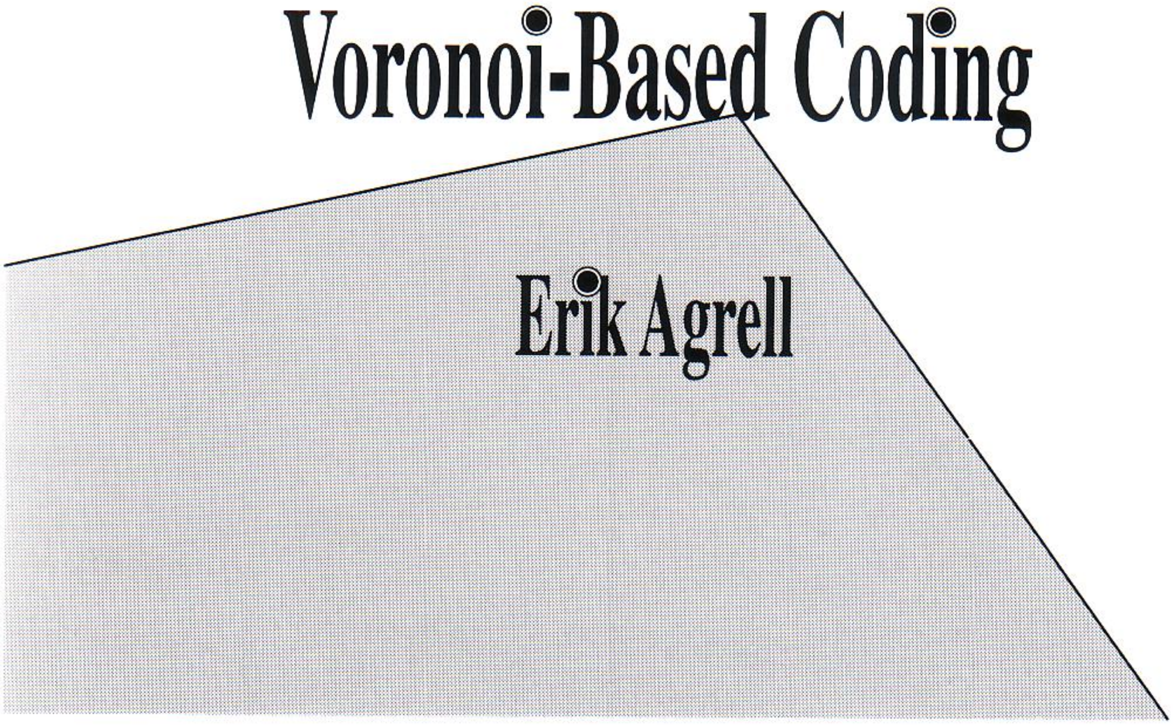
Voronoi-Based Coding



Erik Agrell

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

GÖTEBORG, SWEDEN

Dissertation for the degree of Doctor of Philosophy

Voronoi-Based Coding

Erik Agrell

Chalmers University of Technology / Chalmers Tekniska Högskola

Göteborg, March 1997

Agrell, Erik
Voronoi-based coding
ISBN 91-7197-464-4
ISSN 0346-718X
School of Electrical and Computer Engineering,
Technical report 303

Front cover: A set of three points and the Voronoi region of one of them.

Back cover: A generator matrix of the best known nine-dimensional lattice for quantization. See p. 92.

Department of Information Theory
Chalmers University of Technology
S-412 96 Göteborg
Sweden

Printed by Chalmers Reproservice
Göteborg, March 1997

***Abstract*—**The performance of a digital communication system can generally be improved by increasing the number of variables being jointly coded. In this sense, it is desirable to have, e.g., higher-dimensional quantizers, longer channel codes, and more users in a multiple-access system. However, increasing the number of variables results in higher complexity of encoding and decoding, which are two limiting factors in the choice of coding methods. In this dissertation, which comprises seven published or submitted articles, algorithms are discussed for three related applications in telecommunications: vector quantizer encoding, multiuser detection, and soft-decision channel decoding. A unified approach is obtained through a common formulation in terms of discrete optimization, or, taking on a geometric viewpoint, searching in multidimensional point sets. A convenient instrument to characterize the structure of these sets is the Voronoi diagram.

The first considered application is vector quantization, with some focus on lattices. Lattices are regular structures of infinitely many points that after truncation can be employed as quantizers. An algorithm is developed to optimize the quantization performance of lattices. Employing the algorithm, two lattices are found that improve on previous results. It is also discovered that the so-called D^+ tessellation, which is a union of two lattices, is superior to any known single lattice in dimensions 7 and 9. Truncated lattices are also analyzed, revealing that their distortion in relation to that of optimal quantization increases with the number of points. An algorithm for quantizer design is introduced that maintains close to minimal distortion by suitably stretching the truncated lattice. The work on vector quantization also includes theory and algorithms for index assignment, which is a way to incorporate error-robustness into the quantizer design through a careful codevector ordering. Another contribution is a comparison of the complexity of some encoding methods.

Multiuser detection in CDMA systems is formulated as the geometric problem of searching a point set having a certain structure. Properties of Voronoi diagrams of such structures are given, thus making it possible to apply a known Voronoi-based search algorithm for detection.

Soft-decision decoding of block channel codes is the third application being studied, again by means of the Voronoi diagram. A fast algorithm is presented to investigate the Voronoi diagram of binary linear block codes. Several well-known codes are characterized. Asymptotic theory shows that for most classes of long codes, the complexity of the Voronoi diagram as a function of the bite rate displays a distinct threshold at the rate of one half. Voronoi-based soft decision decoding is essentially practicable only for rates above this value.

***Index Terms*—**Source coding, channel coding, Voronoi diagram, nearest neighbor search algorithm, complexity, vector quantization, lattice, index assignment, linear block code, soft-decision decoding, Gaussian channel, neighbor descent, asymptotic theory, computational geometry.

Preface

Before diving into the details of telecommunications, some general comments on this work may be appropriate. For the preface to be accessible by a wide audience, the presentation follows a different style here than in the rest of the dissertation. Depending on your background, you may prefer one of the styles to the other. If you feel offended by a lack of scientific formalism and precision, please stop reading now and skip to page 1.

To begin with, let us make sure that everybody understands the title. When you want to transmit a message somewhere, you are not always so lucky that the available medium, or channel, is suited for the type of message. For example, a telephone wire cannot accommodate sound, only voltage. Or a movie is not visible very far through the atmosphere, but electromagnetic radiation (of suitable wavelength) can propagate almost forever. Hence, it is a good idea to include some *coding* in a telecommunication system. Coding is the translation of a signal from one form to another, such as the representation of your voice as a time-varying voltage, or the image of Ingrid Bergman as radio waves.

Voronoi-based refers to a certain geometric idea. To visualize it, image a large garden between your house and your neighbor's. Both you and your neighbor love strolling in the garden and picking the wonderful flowers. However, the neighbor sometimes intrudes on your half of the garden, picking flowers that grow noticeably closer to your house than to his, so one day you build a fence, exactly halfway between the two houses. From that day you never see your neighbor again. Encouraged by the success, you build similar fences to separate you from your neighbors in other directions from your house, too. *Yes, I know, this is a tale of human tragedy, but I need the scenario to explain some geometry. When I am done, you may tear the fences down and invite all your neighbors for coffee.* Your barrier now encloses all the flowers that grow closer to your house than to anyone else's. This is the *Voronoi region* of your house. If all of the inhabitants in the neighborhood would build fences halfway between each other, the *Voronoi diagram* of the houses would be made visible. It certainly takes some time to build all these fences, but once it has been done, it greatly simplifies the distribution of any future flower. If you skim through this book, you will discover several Voronoi diagrams. There is even a Voronoi region on the front cover.

By now, I hope that any nonspecialist reading this has at least a vague idea of what "Voronoi-based" and "coding" mean. What remains for me is to explain what the two concepts have to do with each other. This is what the rest of the book is about. Basically, some problems in coding theory are analyzed using the Voronoi diagram as a mathematical tool. *My apologies if you were led to believe that this is a book on gardening. You will be awfully disappointed.*

By the way, one more thing about Voronoi diagrams, and about geometry in general. What I described above is a two-dimensional example, Voronoi diagrams in the plane, but the concept can be directly generalized to more dimensions. For instance, the same story as above could be told of a group of birds in the jungle, living not only in different trees but also on different heights. If they decide that every mosquito belongs to the bird in the nearest nest, and build cages around their territories, they create a three-dimensional Voronoi diagram. *I suspect that birds do not normally build cages around themselves, but again, this preface was never expected to satisfy notorious scientists.* Four dimensions is where human common sense normally breaks down. Whether this limit is physical or perceptual is an interesting question, unfortunately beyond the narrow scope of this boringly technical dissertation. The important point is that higher-dimensional structures can still be described mathematically, and we should not let our inability to visualize them prevent their employment in coding.

The efforts that have led to this dissertation would have been absolutely impossible without help. Qualified help, in numerous matters. In this context, I feel incapable of mentioning anyone before my family. Magdalena, if this book had had a dedication, it would have been to you, “who provided me with two essential ingredients—Time and Love,” but in the first place, dissertations do not normally carry printed dedications, and in the second place, Toby Berger used this formulation in 1971. I believe I had better say what needs to be said in person instead. Alfred, you have given me a taste of what happiness might really be about. I admire your ability to find it, and also to give it. Your crystal-clear sense of logic and your unprejudiced way of observing the world invigorate our conversations greatly. I have a lot to learn from you. Little Robert, I do not know you very well yet, but already you mean so much to me. When my burden feels heavy, I think of you and perceive a vision of brighter days to come. You represent the future.

My colleagues at the Department of Information Theory deserve a vector of thanks. The first acknowledgement goes to Per Hedelin, my supervisor, who introduced me to the field of information theory and showed me the fascination (indeed, the beauty) of vector quantization. Regardless of what I might work with in the future, I will never forget our venture into this realm. Per has the ability to find unexpected solutions, of research problems as well as of practical ones. Once when I intended to write an article, I just did not know how to begin. The research was done, I was satisfied with my results, but I felt unable to formulate a suitable line of approach. I struggled with it for days, producing exactly as much text as I rejected. Per though about the problem for a moment, then he told me to write the last chapter first. I did, and then everything else fell neatly into place. His piece of advice is one that I wish to share with all of you.

I have had the pleasure to cooperate with several inspiring colleagues. In addition to Per, I worked with Thomas Eriksson, Petter Knagenhjelm, and Tony Ottosson, and I learned different things from each of them. Thomas is gifted with an incredibly persistent optimism. Thank you for your stubborn belief that even our most imaginative ideas would be fruitful! Sometimes you were actually right, which resulted in articles 3 and 4. Petter’s ability to present a complicated topic in a comparatively comprehensible way has been an inspiration for me, ever since I was an undergraduate student. Also, Petter is one of the most generous persons I know, in the widest meaning of the word. Tony has a higher ratio of things-that-he-does to things-that-he-says-should-be-done than most people. By the way, this merit is not due to any restraint on the amount of talking.

Mikael Skoglund and Stefan Dodunekov have been very helpful as discussion partners on various subjects. I have always felt welcome to ask them when in need of, say, a certain formula or reference. Their patience with my technical and mathematical shortcomings and their endurance with my most elementary questions is gratefully acknowledged.

Ingvar Jönsson, my friend, you always supported me. I appreciate your hearty encouragement and I preserve many happy memories from our years of teaching Digital Radio Communications together. Ingvar has taken a keen interest in my work on block codes, and I especially want to mention the so-called “H rule” in article 7, which was inspired by a suggestion of Ingvar.

In the completion of this dissertation, I have paid less attention to other matters than I should. Thanks to everyone who reduced my nondissertation duties, both at work and at home! My fellow Ph.D. students have been very kind to me, and so have my family, and Magdalena’s. Without your help, my task would have been hopeless.

I am indebted to Per, Tony, Petter, Mikael, and Thomas for careful and competent proofreading. The practical assistance of Lars Kollberg and Eva Axelsson has also been much appreciated. And, not to forget, I have often sent a grateful thought to all those fabulous cookie makers, who provided me with the physical energy source essential for extensive periods of continuous work.

Now fasten your seat belts and join me for an exciting tour in the city of Telecommunications!

Contents

Introduction

1 E. Agrell and P. Hedelin, “How to evaluate search methods for vector quantization,” in *Proc. Nordic Signal Processing Symposium*, pp. 258–263, Ålesund, Norway, June 1994.

2 P. Knagenhjelm and E. Agrell, “The Hadamard transform—a tool for index assignment,” *IEEE Trans. Inform. Theory*, vol. 42, no. 4, pp. 1139–1151, July 1996.

3 E. Agrell and T. Eriksson, “Optimization of lattices for quantization,” submitted to *IEEE Trans. Inform. Theory,* Dec. 1996.

4 T. Eriksson and E. Agrell, “Lattice-based quantization, part II,” Tech. Report 18, Dept. of Information Theory, Chalmers Univ. of Technology, Göteborg, Sweden, Oct. 1996.

5 E. Agrell and T. Ottosson, “ML optimal CDMA multiuser receiver,” *Electronics Letters*, vol. 31, no. 18, pp. 1554–1555, Aug. 1995.

6 E. Agrell, “Voronoi regions for binary linear block codes,” *IEEE Trans. Inform. Theory*, vol. 42, no. 1, pp. 310–316, Jan. 1996.

7 E. Agrell, “On the complexity of soft-decision decoding,” submitted to *IEEE Trans. Inform. Theory,* Jan. 1997.

Articles 1, 2, 5, and 6 have passed a review.

“There is a curious and provocative duality between the properties of a source with a distortion measure and those of a channel.”
Claude Elwood Shannon, 1959 [81]

Introduction

1. The Telecommunication System

The purpose of this introduction is to illuminate the results of the seven included articles in a broad perspective, to outline their connections with each other and with related work. A suitable point of departure is the digital communication system in figure 1. In this section, the system is described from the user’s point of view; subsequent sections will go into more technical and mathematical detail in order to specify the problems that are considered in the articles.

The diagram in this figure has become something of a standard model and it is reproduced in numerous textbooks in information theory and digital communications. Strangely, no one seems to know who first pictured it. An early appearance is in [72, p. 2]. The model is quite general and can represent a wide range of digital telecommunication applications, including both transmission and storage. It does not,

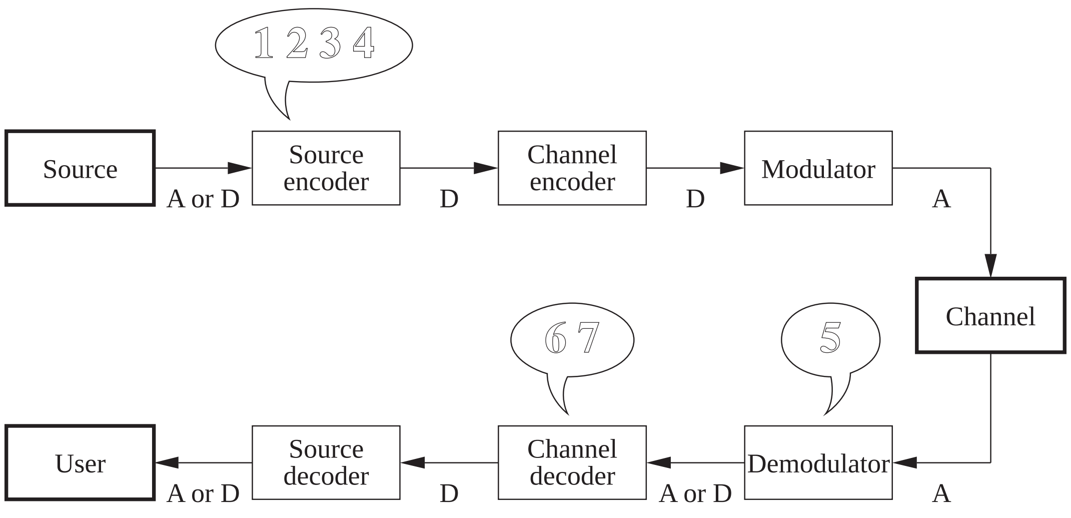


Figure 1. The communication system. Letters indicate if a signal is analog or digital, and numbers refer to the articles.

however, cover every imaginable system.

Suppose that we need to communicate a message somewhere, such as speech, an image, a video sequence, medical or meteorological data, or music. These are examples of analog messages; among digital messages, we can mention text and computer programs. The origin of the message is called the *source.* The (physical) *channel* is any medium in which a signal can be transmitted or stored, such as a radio frequency band, an electric wire, an optic fiber, a magnetic disk or tape, or a CD. Most channels have in common that they are analog by nature, and that they are more or less noisy, in the sense that they distort the signal. The *user,* at the end of the communication chain, is the one for which the message is intended. It may be you or me, or a computer.

The source, the channel, and the user are to be regarded as fixed for the communication theorist, whose object is to design a coding scheme matched to these three blocks. This is the purpose of the other components in figure 1. The design aims to present the user with as accurate a message as possible, while occupying a minimum of channel resources. The two demands conflict with each other, which calls for a compromise.

The object of the *source encoder* is to represent the source as a sequence of bits (or symbols), in such a way that the *source decoder,* when fed with the same sequence, is able to reproduce the message faithfully. If the source is analog, the source encoder includes *sampling,* to produce a discrete-time signal, and *quantization,* to make the signal discrete in amplitude as well. This dissertation has a lot to say on the subject of quantization, beginning in section 4 and then continuing in articles 1–4. In source coding, the compromise mentioned in the previous paragraph takes the shape of a tradeoff between accuracy and the number of bits in the representation.

The communication system is traditionally designed under the assumption that the source decoder receives a bit sequence that is identical to the output of the source encoder. To increase the probability that this will indeed be the case, despite the hazard that a noisy channel poses, the *channel encoder* adds some redundant bits, which the *channel decoder* utilizes to detect if some bits have been corrupted, and possibly also to correct the errors.

As mentioned above, physical channels have normally an analog nature: they carry waveforms, such as a time-varying voltage or a configuration of matter. To represent bits in such a medium, a digital *modulator* is employed, which maps each bit or group of bits into a certain waveform. The *demodulator* performs the reverse mapping while neutralizing as much as possible of the distortion induced by the channel, e.g., it combats noise with filtering, dispersion with equalization, and multiple access interference with interference cancellation.

If the source encoder, channel encoder, and the modulator are combined into one general “transmitter,” and the corresponding operation is done on the receiver side, we obtain Shannon’s classical model, from which essentially all communication models today can trace their origin. Shannon’s paper in 1948 [80] is commonly recognized as the single most important paper that has ever been published in the field of information theory. He defined the *entropy* of a digital source and the *capacity* of a channel; furthermore, he showed that for error-free transmission, the source entropy is upperbounded by the channel capacity [13], [23]. Since then, numerous divisions of the transmitter and receiver into smaller units have been presented, depending on practical as well as pedagogic preferences. The structure that is illustrated in figure 1 is sufficient for the purpose of this introduction.

One consequence of Shannon’s proof of his famous theorem mentioned above is that the upper bound is attainable through a separate design of what is now known as a source coder and a channel coder. However, Shannon himself did not use the terms source and channel coding,[1] nor did he suggest the separation to be built into practical communication systems. The drawback of source-channel separation is, from an engineer’s point of view, that it is optimal in an asymptotic sense, as the complexity and the delay of the encoder and decoder are allowed to increase towards infinity [89], which is clearly not a practical assumption. Nevertheless, most efforts in communication theory since the sixties have involved separation. Only recently has the notion of what is nowadays called combined (or joint) source-channel coding seen a renaissance. A recent reference discussing this topic is [57]; see also section 4.

To zoom in on the work that is contained in this dissertation, consider figure 1 again, and note especially where an analog signal is converted into a digital one. This conversion occurs twice in a digital communication system, once in the transmitter and once in the receiver. In the transmitter, the source encoder has to find out which digital representation, out of a finite number of available ones, that is most accurate for a given analog source signal. The receiver, which observes an analog signal on the channel, has knowledge of the finite set of signals that can possibly have been transmitted; it should decide which one of these that underlies the observation, taking channel distortion into consideration. This decision process can be carried out in either the demodulator or the channel decoder, depending on the system design.

The two digital-to-analog conversion processes can be described as instances of the same search problem: Find the signal in a prespecified set that is closest, in some well-defined sense, to a given signal. To facilitate a unified approach to the compo-

[1] The common usage of the terms appears to emanate from Fano’s book in 1961 [27].

nents of the communication system where analog-to-digital conversion occurs, we first pause to define and study the search problem from a geometric point of view. This leads into the important concepts of *nearest neighbor search* and *Voronoi diagrams,* which are the topics of the following two sections, 2 and 3. For the sake of a unified approach, the basic theory is introduced from a “neutral” point of view, i.e., without any specific application in mind. Beginning in section 4, the discussion returns to telecommunications, and especially to the source encoder, the demodulator, and the channel decoder. These are the blocks in figure 1 where the search problem arises. Each of the blocks has its own section (4–6), where the connection between coding and geometry is established, and the positions in the system of the seven constituent articles are clarified.

2. Voronoi Diagrams

The Voronoi diagram plays a central role in this dissertation. In short, it represents a classification of the points in Euclidean space according to which one of a number of reference points that they are closest to. This section will initially define the Voronoi diagram and related concepts of geometry such as the nearest neighbor search problem, and then proceed into a more algorithmically oriented discussion of how the Voronoi diagram can be determined. Subsequent sections will discuss how it can shed light on some coding problems.

Assume that a countable set $\mathcal{C}$ of d-dimensional vectors is given, and that another d-dimensional vector $\mathbf{x}$ is observed.[2] The task of finding the vector in $\mathcal{C}$ whose Euclidean distance to $\mathbf{x}$ is minimal is called *nearest neighbor search,* or the closest point problem, or sometimes the post-office problem. We introduce the function $\mathbf{n}$ to denote the nearest neighbor, defined as

$$\mathbf{n}_{\mathcal{C}}(\mathbf{x}) = \arg\min_{\mathbf{c}\in\mathcal{C}} \|\mathbf{x}-\mathbf{c}\|^2 \tag{1}$$

where $\|\cdot\|^2$ denotes the squared Euclidean norm of a vector. This function associates every d-dimensional vector with a vector in $\mathcal{C}$; the set of *all* d-dimensional vectors being associated with the same vector is called the *Voronoi region* of that vector. Formally, the Voronoi region of $\mathbf{c}\in\mathcal{C}$ is $\{\mathbf{x}\in\mathbb{R}^d : \ \mathbf{n}_{\mathcal{C}}(\mathbf{x}) = \mathbf{c}\}$. The set of Voronoi regions of all points in $\mathcal{C}$ cover the space completely, overlapping each other only at the boundary points of the regions.[3] This set of regions, and also a graphical illustra-

[2] In order to maintain an interdisciplinary treatment in this section, we avoid naming the variables $\mathcal{C}$ and $\mathbf{x}$.

[3] Provided that $\mathbf{n}$, in cases where (3) results in a tie, is allowed to be multiple-valued.

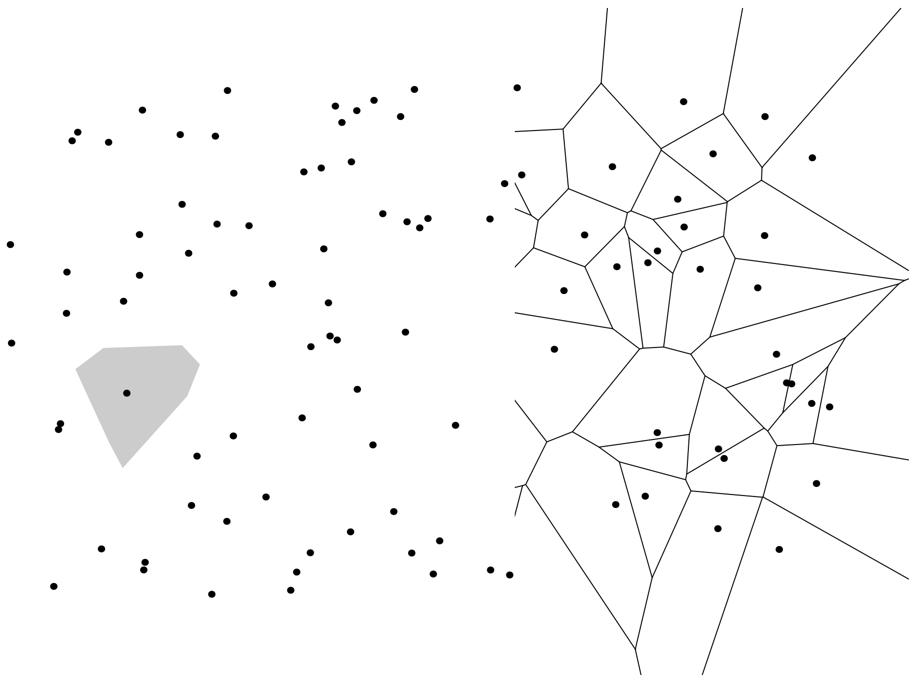


Figure 2. Features of a point set. The points themselves (dots), a Voronoi region (shaded), and a part of the Voronoi diagram (lines).

tion thereof, is called a *Voronoi diagram*. See figure 2 for an example. Another Voronoi region is the shaded region on the front cover.

Since so much literature has been devoted to theory and methods specific for two-dimensional applications, it should already at this point be emphasized that no major concepts or results in this dissertation are theoretically restricted to two dimensions. However, most of the illustrations are, for nothing but graphical reasons, two-dimensional. This can actually be quite misleading sometimes, especially when the purpose of the figure is to illustrate a phenomenon whose significance increases with the dimension. The reader is encouraged to generalize the ideas behind the illustrations mentally, at least to three dimensions.

A Voronoi diagram is, as subsequent sections will show, a useful tool in several problems of communication theory. However, its use is not at all limited to communications; it has been employed, and also reinvented, in a remarkably large number of seemingly unrelated applications, ranging far outside the field of engineering. In fact, the main textbook on the subject of Voronoi diagrams lists 19 sciences or disciplines where applications have been found [69, p. 2], but none of them covers the coding problems that are considered in this dissertation.[4]

The importance of this geometric structure is apparent from the number of names that have been used for it. Some terms that have been given for "Voronoi region," or something very similar, are: area of influence polygon, area potentially available, Blum's transform, Brillouin zone, capillary domain, decision region, Dirichlet

[4] However, analog-to-digital conversion is briefly mentioned on pages 343 and 428 in the book.

region, domain, domain of action, domain of influence, honeycomb, maximum likelihood region, nearest neighbor region, plant polygon, plesiohedron, proximal polygon, region, representation region, Thiessen polygon, tile, Wabenzelle, Wigner-Seitz cell, Wirkungsbereich, and some variations thereof [69, pp. 7–8, 341–343, 369, 376–377], [26], [39, pp. 89, 169], [7], [20, p. 33], [79], [38], [83], [88, p. 48], [77]. It appears impossible to determine who first introduced Voronoi diagrams, but we know for sure that the concept is far older than any of the persons whose names have been given to it. Similar diagrams were used in the 17th century, according to preserved documents [69, pp. 6–7], but nothing indicates that the idea should have been a novelty at the time.

To determine the Voronoi diagram, or its geometric dual, which is the *Delaunay triangulation*, is computationally costly in high dimensions. Several methods have been suggested. In 1979, Brown discovered a mapping such that the Voronoi diagram of a d-dimensional point set is transformed into the convex hull of a $(d+1)$-dimensional point set [15]. Hence, any general convex hull algorithm would imply a method to construct Voronoi diagrams. This discovery has become more significant today than it was at the time of its publication, because of the progress that was made during the 80's in the computation of convex hulls. For recent overviews of convex hull algorithms, see [25, ch. 8], and [69, pp. 260–261].

Bowyer [14] and Watson [95] in two similar papers in 1981 both described how a Voronoi diagram is modified by the insertion of a new point into the set, thus obtaining incremental algorithms for the construction. A different approach was suggested by Gersho in 1982 [29], who employed a Monte Carlo technique to identify the shape of the Voronoi regions using a training set. The probability that this method finds the true Voronoi diagram tends to 1 as the size of the training set approaches infinity; for moderate training set sizes, an approximation of the Voronoi diagram is obtained. The accuracy of the approximation can be traded for execution speed.

In 1983, Avis and Bhattacharya showed that the Voronoi diagram for a given set of points can be established through the solution of a large number of linear programming problems [8]. This line of thought was continued by Joshi and Poonacha [47], who with a slightly different formulation employed the feasibility problem in linear programming, and Agrell [2], who used a similar technique to characterize some other properties of a Voronoi diagram. Hartvigsen employed linear programming to the reverse problem, that of determining whether a given partition is a Voronoi diagram of some point set and, if so, finding this point set [41].

So far, the discussion has been concerned with arbitrary point sets. In practical problems, however, the point sets that appear are often constrained to follow a certain structure, and corresponding constraints apply to the Voronoi diagrams. Such cases will be discussed in connection with some coding problems in sections 4–6 and articles 3–7. It turns out that the general algorithms summarized above may not be suitable for computing the Voronoi diagram of a structured point set. They may be prohibitively slow, and they may even fail to yield correct results due to the degenerate nature of the structure. We will see examples of both cases later in the dissertation.

3. Nearest Neighbor Search Methods

We now turn to another problem of profound practical significance in communications: nearest neighbor search. Implicit in the definition of $\mathbf{n}_{\mathcal{C}}(\mathbf{x})$, see (1), lies a method to evaluate the function: compute the distance to all points in $\mathcal{C}$ and select the point that yields the minimum distance. This method, *full search,* is conceptually the simplest way of performing nearest neighbor search, but it is also one of the slowest. During the past two decades or so, considerable efforts have been devoted to faster search methods for the purpose.

Since the Voronoi diagram can be seen as a representation of the combined solutions of all possible nearest neighbor search problems for a given point set, it is not surprising that the structure, once determined and stored in a suitable form, can be utilized in nearest neighbor search. To begin with, if we want to assess whether a certain point $\mathbf{c} \in \mathcal{C}$ is the nearest neighbor $\mathbf{n}_{\mathcal{C}}(\mathbf{x})$ of a given vector $\mathbf{x}$, it is sufficient to compute the distances from $\mathbf{x}$ to $\mathbf{c}$ and to all the *neighbors* of $\mathbf{c}$. (A neighbor[5] of a point $\mathbf{c} \in \mathcal{C}$ is another point in $\mathcal{C}$ whose Voronoi region shares a $(d-1)$-dimensional facet with the Voronoi region of $\mathbf{c}$.) If $\|\mathbf{x}-\mathbf{c}\|^2$ is the shortest of these distances, then $\mathbf{n}_{\mathcal{C}}(\mathbf{x}) = \mathbf{c}$ and no more distances need to be computed. On the other hand, if $\mathbf{x}$ is closer to one of the neighbors of $\mathbf{c}$ than to $\mathbf{c}$ itself, then this neighbor can replace $\mathbf{c}$ in a new test of the same kind. Repeating this procedure yields an iterative nearest neighbor search algorithm, beginning at an arbitrary point in $\mathcal{C}$ and terminating at $\mathbf{n}_{\mathcal{C}}(\mathbf{x})$, without computing all the distances in (1). The algorithm, which we call *neighbor descent* because of its resemblance to steepest descent methods for the minimization of continuous functions, is illustrated in figure 3. It assumes that the neighbors of all points in $\mathcal{C}$ have been determined in advance and stored, which for some sets may require a considerable amount of memory. Neighbor descent appears, in one form or the other, in articles 1 and 4–7.

This search method was initially suggested by Green and Sibson in 1978 [38], [82], who used it as an element in the construction of two-dimensional Voronoi dia-

[5] A (Voronoi) neighbor is also called an adjacent, a contiguous, or a relevant vector. Note that the word “neighbor” in “nearest neighbor search” does not refer to Voronoi neighbors.

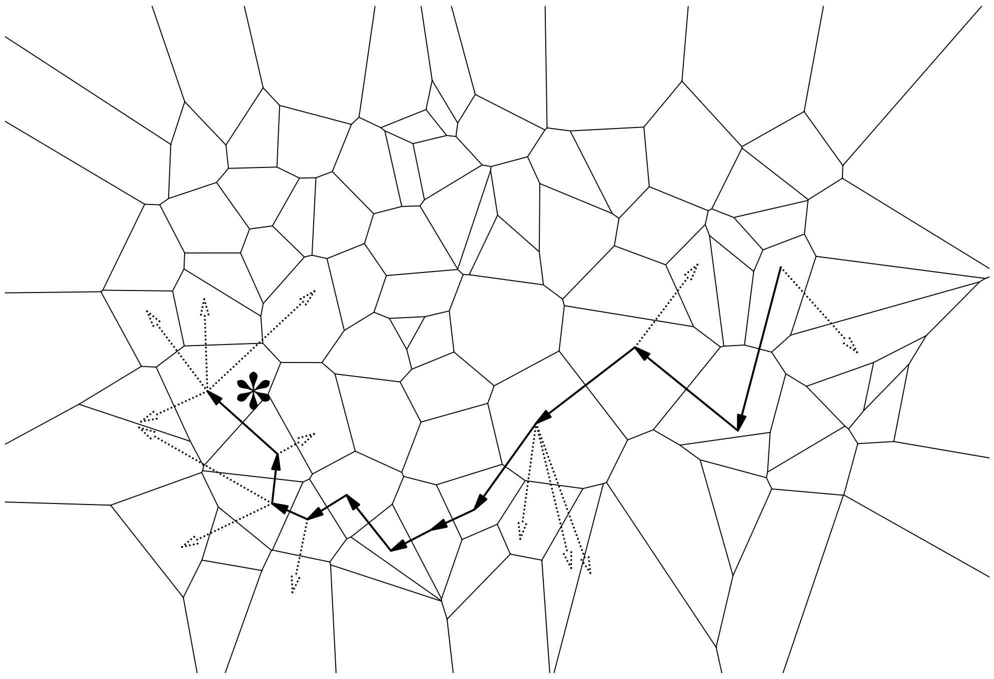


Figure 3. An instance of the neighbor descent search method. The position of the vector $\mathbf{x}$ is marked with a star.

grams. Ohya *et al.* [68] and Okabe *et al.* [69, pp. 225–230] cover this use of neighbor descent thoroughly. The method was applied by Hwang [44] and Butovitsch [16, pts. D–E] to channel decoding, see section 6, and by Joshi and Poonacha to vector quantization [47].

Several variants of the neighbor descent method come to mind. If more than one neighbor would decrease the distance to $\mathbf{x}$, it is not evident which one should be selected. Agrell compared three strategies,[6] and showed that the most efficient choice is the *first* favorable neighbor found, which is not necessarily the same as the *best* one [3]. Another way to vary neighbor descent is to use a different "neighbor" concept, one that is not based on the Voronoi diagram. Such approaches were suggested by Hwang and Butovitsch in their above-mentioned publications, and also by Arya *et al.* [4], [5], [6] and Jeong and Gibson [45]. Some results in this area are included in articles 1 and 4. It is important to remember, however, that when the Voronoi neighbors are abandoned, the neighbor descent method is not anymore certain to find $\mathbf{n}_{\mathcal{C}}(\mathbf{x})$; it might return a suboptimal point instead.

Neighbor descent is not the only alternative to full search, when it comes to performing nearest neighbor search. On the contrary, a tremendous amount of work has been done on search algorithms in the last two decades. Some of them are guaranteed to return $\mathbf{n}_{\mathcal{C}}(\mathbf{x})$, whereas other algorithms find suboptimal solutions of the search

[6] In an unpublished report, they were characterized as, respectively, timid ("let's not do anything until we know what the choices are"), avid ("get anything that is better than before"), and rigid ("since this has worked so far, it is worth trying once more").

problem. In one sense, the field has begun to stagnate: not in the amount of literature issued every year, but perhaps in the relation between such literature and previous work. The wheel is continually being reinvented. The disorder has reached a stage where the presentation of another algorithm would almost certainly disappear behind the heap of similar work. Hence, an important contribution to nearest neighbor research would be to somehow sort the items in that heap, thus making it easier for future authors to put their contributions on top of it. In some other research areas where a similar problem has occurred, a textbook has appeared and resolved the confusion. This has not yet happened in this field; the closest thing to a textbook is a couple of dissertations [60], [18], [75], [4].

Several authors have compiled literature surveys of the main approaches to nearest neighbor search. Their surveys cover to some extent different parts of the literature and use different subdivisions; together they constitute a fairly good overview of the field. Instead of providing another (incomplete) survey of algorithms, I will in this introduction give a short survey of surveys, which, to my best knowledge, has not been done before.

The book by Gersho and Gray contains a comprehensible summary of some main categories of nearest neighbor search methods [31, pp. 332–335, 479–481], without going into detail on individual algorithms. For individual algorithms, the surveys in the dissertations by Cheng [18, pp. 14–22] and Arya [4, pp. 3–8] are good sources of information. The surveys by Vidal *et al.* [90], [91] mention a large number of references, perhaps larger than any other survey does, but the presentation is brief. The earliest work in the field is surveyed in the well-known paper by Friedman *et al.* [28]. Ramasubramanian and Paliwal give a survey that concentrates on the development in the so-called k-d tree search methods [76], and the one by Moayeri and Neuhoff in [61] covers mostly the same material. In addition, short but enlightening summaries are included in [17], [78], [9], and [51].

One reason for the multitude of nearest neighbor search algorithms that have been proposed (at least a few hundreds, reinventions included), and for the absence of a consensus regarding which algorithm is actually the “best,“ is that the quality of a search algorithm can be measured in so many ways. This problem is the topic of article 1, which is further discussed in the next section.

4. SOURCE ENCODING

In this and the two following sections, the discussion is brought back to where it began, the communication system illustrated in figure 1. The three components where analog-to-digital conversion occurs will be highlighted in one section each, and various coding problems will be formulated in terms of nearest neighbor search.

Source encoding implies the representation of a message as a sequence of bits. How this is done depends, obviously, on the type of source and application. For efficient communication, different types of sources need their own source coders. A typical encoder operates in two steps; first it creates a parametric representation of the signal, then it quantizes the parameters.

The selection of a set of parameters for a certain type of source is a vast research field in itself, divided into speech coding, image coding, etc. The parameters should identify the significant features of the source signal. It is unnecessary, even wasteful, to adopt parameters that contain too exact a description of the signal, if some details are more relevant to the user than others, which is certainly the case in the above-mentioned applications.

To complete the digital representation, the parameters are quantized, one at a time (*scalar quantization*) or several (*vector quantization*). Because of its connection with search problems and Voronoi diagrams, we concentrate on vector quantization. Several tutorials and overviews have been written in this field. All-round treatments of vector quantization are [36], [84], [37, ch. 5], and [31], whereas [35] and [12] concentrate on a certain type of quantizer. Applications in speech coding are surveyed in [30], [53], [71], [32], and [19], and in image coding in [66], [74, ch. 12], [22], and [21].

The core of a vector quantizer (VQ) is the *codebook* $\mathcal{C}$, which determines the set of parameter vectors that can be represented by the coder, i.e., the *codevectors*. This set is a finite list, so the output of the source encoder is simply an integer, most often coded as a sequence of bits, which gives the index of the selected codevector in the codebook.[7] The source decoder maintains its own copy of $\mathcal{C}$ and identifies the codevector whose index equals the received integer. If the number of codevectors in $\mathcal{C}$ is 2^k, and each vector contains d elements, then the *rate* k/d gives the number of bits that are used to encode one parameter.

The vector quantizer is presented with a search problem. After receiving a parameter vector $\mathbf{x}$, its task is to find a vector $\hat{\mathbf{x}}$ in $\mathcal{C}$ that is similar to $\mathbf{x}$, in some sense. How to measure similarity is a delicate question, whose answer in the end depends on the sensitivity of the user to various kinds of changes in the message and on the sensitivity of the message to changes in the parameter set. These sensitivities are hard to assess, and even harder to express as a mathematical function of $\mathbf{x}$ and $\hat{\mathbf{x}}$. To avoid these difficulties, a reverse approach is often preferred: First choose a distor-

[7] It is assumed that the computer represents the set $\mathcal{C}$ as a list, which is why a codevector can be represented by its index.

tion measure, and then transform the parameter vectors into a coordinate system where this measure is not too inappropriate. By far the most common distortion measure is the squared Euclidean distance, $\|\mathbf{x}-\hat{\mathbf{x}}\|^2$, which corresponds to the energy of the quantization error. Parameters for miscellaneous types of sources have been developed for use with this measure. The main advantage, regarding the complexity of analysis and algorithms, is that the setup creates a nearest neighbor search problem; for minimal distortion, the quantizer should select $\hat{\mathbf{x}} = \mathbf{n}_{\mathcal{C}}(\mathbf{x})$. Consequently, the objective in vector quantizer design for a certain type of parameters is to find the set $\mathcal{C}$ of a given size that minimizes the distortion per parameter

$$D = \frac{1}{d} E\left[\|\mathbf{x}-\mathbf{n}_{\mathcal{C}}(\mathbf{x})\|^2\right]. \tag{2}$$

The evaluation of the expectation assumes that a statistical model is known, or can be estimated, for the parameter vectors to be quantized.

Many of the algorithms that have been proposed for nearest neighbor search were originally presented in the context of vector quantization. Despite the large number of available algorithms—or perhaps because of it—no evident champion has evolved, which leaves anyone building a vector quantizer application with an intricate search problem, namely, the search for a search algorithm. Article 1 is a case study that illustrates why this problem cannot be solved once and for all. Three nearest neighbor search algorithms are evaluated using several performance measures. The considered measures are average search time, worst case search time, storage, precomputation time, and distortion when the search is interrupted after an allotted period of time. In turns out that any algorithm of the three investigated ones emerges as the winner, depending on which measure is used. Geometrically stated, performance is not a scalar but a vector, and to determine the "greatest" vector is generally an ambiguous task. The conclusion is that there is actually a "market" for a multitude of algorithms, and before an algorithm is selected for a certain application, it is important to have precise specifications of the available hardware and the system requirements. As a sideline, which follows naturally from the notion that algorithms have different specialties, the three algorithms are combined into one. This hybrid algorithm outperforms its three constituent algorithms in most aspects, which is illustrated in figure 4. Similar hybrids were discussed in [47], [69, pp. 228–232], [5], and [4].

Let us for a moment return to the source-channel separation theorem, which was summarized in section 1. It implies that the channel and the channel coder can be completely disregarded in the design of a vector quantizer, provided that there is no constraint on the dimension of the parameter vectors or on the codebook size. Such constraints pertain, however, to all practical systems, because of the limited storage,

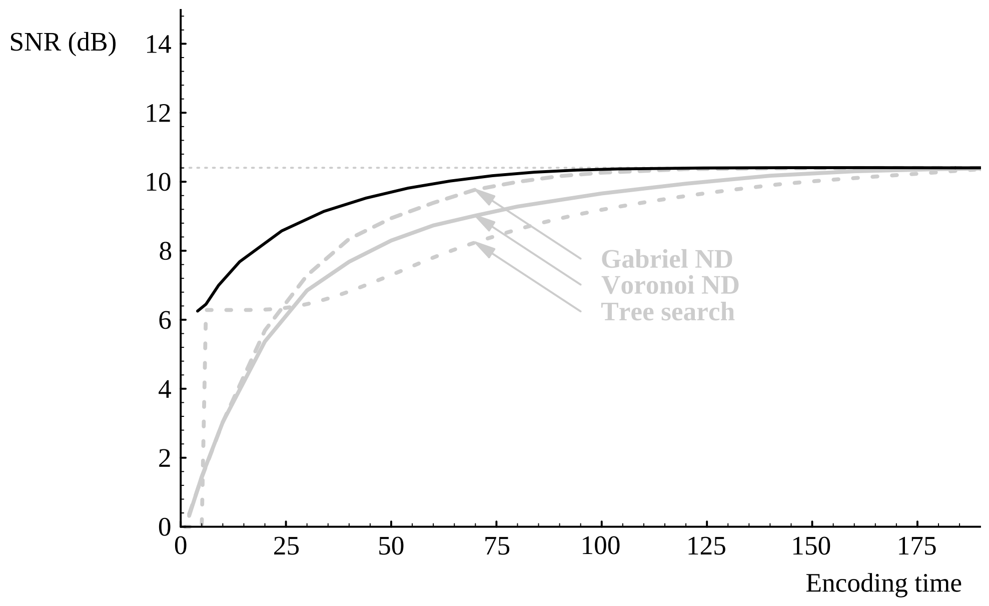


Figure 4. The hybrid algorithm in article 1 finds a good candidate $\hat{\mathbf{x}}$ fast. (From the oral presentation of article 1 at NORSIG -94, but not included in the article itself.)

encoding complexity, and communication delay that can be tolerated. Hence, the relatively small quantizers that can be implemented using today's technology certainly benefit from a design where the possibility of a noisy channel is considered.

If the source decoder always receives the same integer index as the source encoder outputs, which is assumed in conventional design of vector quantizers, then it is irrelevant which index that corresponds to each codevector. This is not the case in a system with channel distortion. If an erroneous index is received, the source decoder will employ the wrong parameter vector. However, the damage can be limited if the incorrect vector is similar to the original one, which is the idea behind *index assignment.* Since a transmitted integer is more easily confused with some integers than others, such index pairs are assigned to codevectors whose coordinates do not differ too much. Typically, the most common error is where just one bit in the binary representation of the index has been inverted. It is worth emphasizing that index assignment is an error-protection method that costs nothing but codebook preprocessing; once carried out, it does not increase the encoding- or decoding complexity, the memory requirements, or the rate. The codebook is stored as a list anyhow; the index assignment is just expressed as a reordering of the entries.

Index assignment can be regarded as a search problem. It is the search for a permutation of 2^k integers such that the average distortion is minimized, assuming a suitable statistical model of the confusion probabilities between indices. The most common model is the *binary symmetric channel,* according to which bit errors occur independently with equal probability. It is enlightening to adopt a geometric approach to this search problem, such that the base-2 representations of the integers from 0 to $2^k - 1$ are interpreted as coordinates in k-dimensional space. The integers thus define the vertices of a hypercube. Index assignment can be regarded as a mapping of these vertices onto the codevectors of a given codebook [49].

The essence of article 2 is that the mapping from the (possibly translated) hypercube to the codebook should be as *linear* as possible. A measure is defined to evaluate how well this goal is achieved, the *linearity*. 2
The search for an index assignment with a high linearity is motivated through theoretical and empirical results, which show that the linearity is closely linked to the distortion, provided that the codevectors are transmitted equally often. For computational purposes, linearity is a more expedient objective than distortion. This is demonstrated through the index assignment algorithm *LISA* (linearity increasing swap algorithm), which in comparison with a few well-known algorithms is shown to achieve a low distortion very fast.

Theoretically, a vector quantizer achieves better performance (lower distortion for a given rate) with an increasing number of parameters d being quantized together [52]. However, to maintain a constant rate, the number of codevectors 2^k increases exponentially with d. This is why so much attention has been given to the nearest neighbor search problem in vector quantization, but still the encoding time is a limiting factor for the codebook size. Storage is another. One way to circumvent the limit is to employ *structured* codebooks. This means that some constraints are imposed on the set of codebooks $\mathcal{C}$ considered in the minimization of (2). It is desirable to select a structure for which a fast nearest neighbor search algorithm and a compact codebook description can be tailored. The attained distortion D of a structured codebook is in general higher than that of the optimal unstructured codebook of the same dimension and rate, but it may well be lower than the optimal codebook of the same *complexity* (search time and memory) and rate, which is a more relevant comparison. Several low-complexity structures have been proposed in the past, sometimes assuming a suboptimal search method: tree-structured VQ, multistage VQ, lattices, block codes, etc. A comprehensive overview of the most common types is provided in [31, ch. 12]. Lattices and block codes are the two types of structured codebooks that are specifically considered in this dissertation.

A *lattice* is formally defined through a generator matrix, $\mathbf{B}$, as the set $\left\{\mathbf{B}^T\mathbf{u} : \mathbf{u} \in \mathbb{Z}^d\right\}$. A more intuitively appealing definition is suggested in figure 5. Since a lattice contains infinitely many points, it is not directly useful as a quantizer, but it can be truncated into a set with a suitable number of points. This raises two questions: How is a good lattice found and how is it shaped into a codebook for a given parameter type?

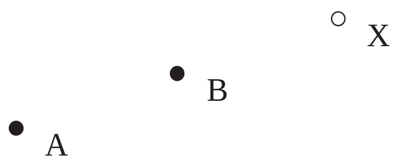


Figure 5. If there are points at A and B, then there is one at X, too. A structure that satisfies this rule for all point pairs is a lattice (or a translation thereof).

The first question is normally addressed by considering the asymptotic problem of vector quantizer design for uniformly distributed parameter vectors, as the rate approaches infinity. This leads to an optimization problem in which the generator matrix is the only variable; boundary effects can be neglected. Much effort has been devoted to this problem, primarily with an algebraic approach [20]. In article 3, we attack the problem through numerical optimization, developing an iterative algorithm to search for good generator matrices for the quantization problem. 3 The algorithm is then let loose in 2–10 dimensions, and for the resulting (numerically represented) lattices, underlying exact expressions are identified. There are indications that we may have reached the optimal lattices in all these dimensions. Most of them are rediscoveries of known lattices, but our 9-dimensional lattice, whose generator matrix decorates the back cover, is of a new type, significantly better than what has been previously reported. We improve on earlier results in 10 dimensions, too. In this case, the obtained lattice has been published before, but not in connection with quantization. Both the 9- and the 10-dimensional lattice may be counterexamples of a long-standing conjecture regarding the connection between this quantization problem and a related lattice problem. Finally, a few cases are shown where the union of two lattices yields a lower distortion than does any single lattice. We believe that the lowest dimension for which this happens is 7.

The second question regarding the design of a lattice-based vector quantizer is how to create a finite codebook, suitable for a given type of parameters, from the lattice. The simplest way is to truncate the lattice, discarding the points outside a certain region [46], which yields a *lattice quantizer.* A theoretical analysis of such a quantizer, assuming a high rate, is provided in article 4. Asymptotically exact expressions for the minimum achievable distortion are derived, 4 which show that the discrepancy between the optimal lattice quantizer and the optimal unstructured quantizer increases with the rate.[8] The explicit expressions were derived for independent and identically distributed Gaussian parameters, but since the shortcoming has to do with forcing a uniform point density onto a nonuniform parameter distribution, we believe that the conclusion can be generalized to a wide

[8] Similar results were independently obtained by Moo and Neuhoff [62].

class of nonuniform distributions. This discouraging result suggests that truncation alone is not sufficient to form a good high-rate quantizer out of a lattice, which motivates our proposition of *lattice-attracted vector quantization.* We allow a lattice to be “stretched,” so that the codebook maintains a small-scale lattice-like structure, while on a larger scale, the point density may vary. The proposed codebook design algorithms attain lower distortions than conventional algorithms under some constrained design conditions, but the main advantage appears when it comes to nearest neighbor search. A variant of the neighbor descent search method is developed for lattice-attracted vector quantizers, where the storage requirements pertinent to ordinary neighbor descent is alleviated by exploiting the underlying lattice structure.

Block codes and their Voronoi regions are discussed in section 6 and in articles 6 and 7. Their traditional application is in channel coding, where soft-decision decoding implies nearest neighbor search. However, block codes have found their use in source coding, too. They were employed directly as low-rate codebooks by Adoul and Lamblin [1] and by Swaszek [85], [86], [87], whereas Hagen obtained codebooks as linear mappings of block codes [40]. The *linear codebook,* which was defined in article 2 as a linear mapping of the vertices of a hypercube, can be regarded as a special case of the latter construction.[9] As pointed out above, there exists a good index assignment for linear codebooks, which makes them relatively robust against channel distortion. A linear codebook with rate one forms the vertices of a parallelepiped. Nearest neighbor search for this structure is an important problem in CDMA demodulation, which is discussed in the next section and in article 5.

5. DEMODULATION

In the conventional form of digital modulation, each bit or group of bits to be transmitted determines one segment of the waveform that is output on the channel [73]. Denoting the possible waveform segments in the time interval $0 \le t < T$ with $s_i(t)$, for $i = 1, \cdots, q$, it is always possible to find a set of orthonormal *basis functions* for the signals $\{s_i(t)\}$. That is, there exist a vector $\boldsymbol{\phi}(t)$ of signals (basis functions) and q vectors $\mathbf{c}_i$ of real numbers, such that $s_i(t) = \mathbf{c}_i^T \boldsymbol{\phi}(t)$ for $0 \le t < T$ and $i = 1, \cdots, q$, and $\int_0^T \boldsymbol{\phi}(t) \boldsymbol{\phi}^T(t) dt = \mathbf{I}$.

This decomposition is utilized in the demodulation. The demodulator contains a bank of matched filters, one for each basis function. If the received waveform is $r(t)$, the output of the filter bank is

[9] Linear codebooks and similar structures have been discussed under the names binary lattice [55], [54], [56], binary residual VQ [12], direct sum codebook [11], LM codebook [43], multistep VQ [36], two-channel conjugate VQ [63], and VSELP codebook [33], [34].

$$\mathbf{x} = \int_0^T r(t)\boldsymbol{\phi}(t)dt. \tag{3}$$

In the ideal case of a channel without any distortion, $r(t) = s_j(t)$ for some j, and it is easily shown that $\mathbf{x} = \mathbf{c}_j$. However, more realistic channel models include some type of distortion, which gives rise to various types of search problems.

For the *Gaussian channel,* the received waveform is $r(t) = s_j(t) + n(t)$, where $n(t)$ is white and Gaussian. Inserting this model into (3), it turns out that the filter bank output is $\mathbf{x} = \mathbf{c}_j + \mathbf{z}$, where $\mathbf{z}$ is a vector of independent, identically distributed, zero-mean Gaussian random variables.[10] Because of the rotational symmetry of the multidimensional Gaussian probability density function, the *maximum likelihood* (ML) hypothesis detector can be implemented with a nearest neighbor search method; the waveform most likely transmitted is the one corresponding to the coefficient vector $\mathbf{c}_l = \mathbf{n}_{\mathcal{C}}(\mathbf{x})$, where $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^q$. The demodulator shows this by transmitting the index l, binary coded.

Commonly used modulation schemes have typically a small value of q, and the point set $\mathcal{C}$ has some regular structure. QPSK, for instance, has its four points $\mathbf{c}_i$ located as the vertices of a square. For structures as simple as this one, there exist trivial search algorithms. More sophisticated search problems arise if other types of distortion than additive noise are included in the channel model, or with other modulation methods.

One such application is *multiuser detection* in a CDMA system [92], [73, chs. 13, 15], [64]. Suppose that the transmitted waveform consists of synchronous contributions from several users, each one BPSK-modulated onto its own *signature waveform,* or spreading code. If the signature waveforms, weighted to account for the transmitted power of each user, are collected in the vector $\mathbf{p}(t)$, then the total transmitted signal is $s_i(t) = \mathbf{b}_i^T\mathbf{p}(t)$. The vector $\mathbf{b}_i$ consists of one bit for each user, represented as ± 1. The number of possible signals is $q = 2^K$, where K is the number of users. To find a set of basis functions for this signal set, let the correlation matrix of the signature waveforms be denoted by $\mathbf{R} = \int_0^T \mathbf{p}(t)\mathbf{p}^T(t)dt$ and let $\mathbf{T}$ be a matrix such that $\mathbf{T}\mathbf{T}^T = \mathbf{R}$, which can be found by Cholesky factorization of $\mathbf{R}$. Now it can be easily verified that the functions $\boldsymbol{\phi}(t) = \mathbf{T}^{-1}\mathbf{p}(t)$ form a basis for $\{s_i(t)\}$, with the coefficient vectors $\mathbf{c}_i = \mathbf{T}^T\mathbf{b}_i$ for $i = 1, \cdots, q$. Hence, ML detection is equivalent to nearest neighbor search in $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^q$. This point set can be visualized as the vertices of a *parallelepiped,* i.e., a tilted hypercube. It is a special case of the "linear codebook" mentioned in the previous section.

[10] The assumed properties of $n(t)$ are physically motivated, see, e.g., [42, pp. 270–273]. However, as long as $n(t)$ is strictly white, its probability density function is irrelevant. That $\mathbf{z}$ is Gaussian still follows from the central limit theorem.

Article 5 develops this geometric treatment of multiuser detection and suggests that maximum likelihood detection, for arbitrary signature waveforms, can be achieved by applying neighbor descent to the point set $\mathcal{C}$. A significant gain in search time is shown, compared to full search. The article also investigates the Voronoi diagram of $\mathcal{C}$. Exploiting properties of the parallelepiped, a fast method to find the Voronoi neighbors is given, and their number is upper-bounded. In [70, ch. 3], the work initiated by article 5 is continued, and results are presented in greater detail. 5

Another channel model worth study is the intersymbol interference channel. Demodulation can in this case, too, be formulated as a nearest neighbor search problem [10], but the topic is not investigated in this dissertation.

The discussion in this section has been confined to demodulators that make an explicit decision on the received signal. It is also possible to postpone this decision, and hence the search problem, to the channel decoder. In this case, the demodulator output simply equals the matched filter output $\mathbf{x}$, which is analog (in amplitude). The next section considers how a channel decoder can benefit from such a system design.

6. Channel Decoding

The purpose of channel coding is error detection and correction. In the channel encoder, redundant bits are added to the sequence, bits that are examined in the channel decoder to assess whether the sequence has been correctly received. Suppose that the output sequence of the source encoder is grouped into blocks of k consecutive bits, and that the channel encoder maps each such block into a block of n bits, where $n \geq k$. Since there are 2^k different input blocks to the channel encoder, its output has an equal number of possibilities. Interpreting the available output blocks, called *codewords,* as vectors, they form a set of 2^k points in n-dimensional space. This set, $\mathcal{C}$, is a *binary block code* and its *rate* is defined as $R = k/n$. Such point sets have been employed in vector quantization, which was discussed in section 4, but their main use is in channel coding.

If the channel is noisy, then what the channel decoder receives may or may not be a codeword. If it is a codeword, then all the decoder has to do is to translate it back to the corresponding k-bit block for further processing by the source decoder. If it is not, the decoder is faced with the problem of estimating which codeword was actually transmitted. Once again, a search problem appears. The nature of this problem can be of two kinds, depending on the type of demodulator.

The demodulator outlined in the previous section performs nearest neighbor search in order to reach a decision on which waveform that was most likely transmitted, and outputs the sequence of bits that represents this waveform. This leaves the channel decoder as a digital-input, digital-output device, which is normally implemented using algebraic methods [50], [96]. An alternative approach, which yields lower distortion at the cost of more complex search problem, is to postpone the search process to the channel decoder. In this case, the analog (in amplitude) output of the matched filters in the demodulator is directly transferred to the channel decoder, which performs *soft-decision decoding.*

Assume for simplicity that binary modulation is employed. The bits are transmitted sequentially, each bit controlling its own waveform segment. Hence, it suffices with only one matched filter in the demodulator. To perform optimal soft-decision decoding, the channel decoder collects n consecutive matched filter outputs in a buffer, corresponding to the n bits of the transmitted codeword, before they are processed. Regarding the buffer contents as a vector $\mathbf{x}$, the search problem in the decoder is to find a codeword $\mathbf{c} \in \mathcal{C}$ that is as similar to $\mathbf{x}$ as possible, where the measure of similarity depends on the employed channel model.

Again we consider the Gaussian channel. With the same method as in the previous section, it can be shown that the vector $\mathbf{x}$ equals $\mathbf{c}_j + \mathbf{z}$, where the two terms are the transmitted codeword $\mathbf{c}_j$ and a vector $\mathbf{z}$ of independent, identically distributed, zero-mean Gaussian random variables.[11] And in this context, too, maximum likelihood detection implies nearest neighbor search; the optimal codeword is $\mathbf{n}_{\mathcal{C}}(\mathbf{x})$. However, the point set $\mathcal{C}$ is quite different from the parallelepiped that appears in CDMA demodulation, and also from the structures considered in section 4, so a study of the Voronoi diagram of $\mathcal{C}$ cannot rely on results from these applications.

The fact that all codewords are binary imposes a special structure on the point set $\mathcal{C}$. It can be regarded as a subset of the vertices of an n-dimensional hypercube. The structure becomes even more attractive for theoretical and computational purposes if we limit the considered codes to being linear as well. With a formulation analogous to the lattice definition in figure 5, a binary block code is *linear* if the modulo-2 addition (exclusive-or) of any two codewords results in another codeword.[12]

Hwang [44], and later Butovitsch [16, pts. D–E], considered the neighbor descent method as an algorithm for soft-decision decoding of binary linear block codes.[13]

[11] Despite the appearance of the expression $\mathbf{x} = \mathbf{c}_j + \mathbf{z}$ in both this section and the previous one, its interpretations differ. In section 5, $\mathbf{x}$ denotes the simultaneous output from several matched filters, whereas here, it means consecutive outputs of the same filter.

[12] Note that “linear” was used in another sense in section 4. The points of a linear *block code* do not in general form a linear *codebook.*

[13] It is remarkable that the same algorithm can be defined in so different ways. Hwang, who employs an algebraic terminology, does not mention Voronoi diagrams at all and refers with “projective set” to

Hwang also found two useful bounds relating the distance between codewords to their being neighbors or not, which for some relatively short high-rate codes yield a full description of the neighbors of all codewords. His paper is concluded with three open questions. The first one, “How do we generally determine the projecting set of a code?” is what article 6 is about. In this article, some geometric properties of the Voronoi diagram are presented, properties that are utilized in the development of a fast algorithm to determine whether two given codewords are neighbors. The new algorithm is then employed to analyze the Voronoi diagrams of some important codes. One conclusion that is drawn from the results is that the coupling between neighbors and their distance is weaker than what was previously assumed [16, pt. D, p. 27].

Decoding algorithms such as neighbor descent, which utilize Voronoi region facets for binary decisions, are efficient only if the number of neighbor pairs in a code is relatively small. Which codes have this property? This is Hwang’s second question, but before going into details on block codes, we elaborate a little on the number of neighbors in a point set as a measure on the complexity of Voronoi diagrams and Voronoi-based algorithms. Binary block codes is by no means the first type of point sets for which the number has been considered. The interest goes back to at least 1897, when Minkowski in his pioneering work on lattices showed that the number of neighbors of a point in any d-dimensional lattice is upperbounded by $2^{d+1}-2$ [59, band 2, pp. 120–121], [58, pp. 81–85, 180–181]. A random lattice reaches this bound with equality, under some general assumptions on the distribution [94, esp., vol. 134, pp. 198–211]. The numbers of neighbors in some specific lattices are given in [93] and [20, pp. 106–135, 456–475]. If no constraint at all is placed on the structure of a point set, it is a remarkable fact that for dimensions 3 and above, the only upper bound on the number of neighbors is the trivial one. This was proved by Dewdney and Vranch [24], who presented an ingenious way to place M three-dimensional points such that all pairs of points are neighbors, regardless of M. Hence, all Voronoi regions for this point set have $M-1$ facets. The result rather dramatically contrasts with the two-dimensional case, where there exists no point set, however large, such that the average number of neighbors exceeds six. Klee obtained some related bounds on the maximum complexity of an unconstrained Voronoi diagram [48]. For randomly generated point sets, low-dimensional results have been analytically derived [65]. The average number of neighbors in a large set of independent uniformly distributed points is 2, 6, 15.54, and 37.78 in dimensions 1 to 4, which suggests that the number of neighbors increases faster with the dimensions

what is here called the set of neighbors of a certain point. The work by Butovitsch follows a geometric style similar to the one used here.

for random point sets than for lattices. Finally, it can be mentioned that even for partitions that are not Voronoi diagrams, such as the tree-structured VQ, which was briefly mentioned in section 4 as one of several possibilities to reduce the complexity of vector quantization, the number of neighbors is a relevant characteristic [67], [97].

Returning to binary linear block codes, a partial answer of Hwang's second question is provided in article 7. The article presents results on the number of neighbors in several well-known codes, computed through a combination of Hwang's bounds, the algorithm of article 6, and a new algorithm. The number of neighbors is observed as a function of the rate R, which reveals an interesting pattern. For codes with $R > 1/2$, a codeword tends to have relatively few neighbors, whereas if $R < 1/2$, most of the codeword pairs are neighbors, and this threshold becomes more distinct with increasing codeword length n. An asymptotic analysis confirms that there is a threshold at $R = 1/2$ for a wide range of codes. In short, the answer to Hwang's second question given by article 7 is "for codes with rates above one half."[14]

Thereby the sightseeing tour in the city of Telecommunications is completed. I have enjoyed being your guide today through the modern architecture of our fast-growing city. As you may have noticed, I was especially delighted to show you the nearest neighbor buildings that have recently been constructed in some of the districts. Did you observe the intricate Voronoi diagrams on their walls? I find them very attractive myself. I hope that you have had a pleasant trip, and wish you a happy stay in the city. If you should choose to venture on your own now into one of the districts, which I highly recommend, remember how easily you get lost. Make sure to bring a good search method!

REFERENCES

[1] J.-P. Adoul and C. Lamblin, "A comparison of some algebraic structures for CELP coding of speech," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 1953–1956, Dallas, TX, Apr. 1987.

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

[3] 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.

[4] S. Arya, "Nearest neighbor searching and applications," Ph.D. dissertation, University of Maryland, College Park, MD, 1993.

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

[6] S. Arya, N. Phamdo, N. Farvardin, and D. Mount, "Fast search algorithms with applications to split and multi-stage vector quantization of speech LSP parameters," in *Proc. IEEE Workshop on Speech Coding for Telecommunications*, pp. 65–66, Sainte-Adèle, Québec, Canada, Oct. 1993.

[14] The third question, "Are there any other algebraic properties which could be used to further reduce the size of the projecting set?" is not discussed in this dissertation.

[7] F. Aurenhammer, “Voronoi diagrams—A survey of a fundamental geometric data structure,” *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, Sept. 1991.

[8] D. Avis and B. K. Bhattacharya, “Algorithms for computing *d*-dimensional Voronoi diagrams and their duals,” in *Advances in Computing Research. Vol. 1: Computational Geometry*, F. P. Preparata, Ed. Greenwich, CT: JAI Press, pp. 159–180, 1983.

[9] S. G. Bakamidis, “An exact fast nearest neighbor identification technique,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. V, pp. 658–661, Minneapolis, MN, Apr. 1993.

[10] L. C. Barbosa, “Maximum likelihood sequence estimators: A geometric view,” *IEEE Trans. Inform. Theory*, vol. 35, no. 2, pp. 419–427, Mar. 1989.

[11] C. F. Barnes and R. L. Frost, “Vector quantizers with direct sum codebooks,” *IEEE Trans. Inform. Theory*, vol. 39, no. 2, pp. 565–580, Mar. 1993.

[12] C. F. Barnes, S. A. Rizvi, and N. M. Nasrabadi, “Advances in residual vector quantization: A review,” *IEEE Transactions on Image Processing*, vol. 5, no. 2, pp. 226–262, Feb. 1996.

[13] T. Berger, *Rate Distortion Theory. A Mathematical Basis for Data Compression*. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[14] A. Bowyer, “Computing Dirichlet tessellations,” *The Computer J.*, vol. 24, no. 2, pp. 162–166, May 1981.

[15] K. Q. Brown, “Voronoi diagrams from convex hulls,” *Information Processing Letters*, vol. 9, no. 5, pp. 223–228, Dec. 1979.

[16] P. Butovitsch, “Classification of signal sets. The classification capability of multi-layer perceptrons and soft-decision decoding of error-correcting codes,” Ph.D. dissertation, Royal Institute of Technology, Stockholm, Sweden, 1994.

[17] S. H. Chen and J. S. Pan, “Fast search algorithm for VQ-based recognition of isolated words,” *IEE Proceedings-I*, vol. 136, no. 6, pp. 391–396, Dec. 1989.

[18] D.-Y. Cheng, “Efficient nearest neighbor search for nonstructured Euclidean codes and application to vector quantization,” Ph.D. dissertation, University of California, Santa Barbara, CA, Apr. 1986.

[19] J. S. Collura, “Vector quantization of linear predictor coefficients,” in *Modern Methods of Speech Processing*, R. P. Ramachandran and R. J. Mammone, Eds. Boston, MA: Kluwer Academic Publishers, pp. 23–50, 1995.

[20] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, 2nd ed. New York, NY: Springer-Verlag, 1993.

[21] P. C. Cosman and R. M. Gray, “Vector quantization of image subbands: A survey,” *IEEE Transactions on Image Processing*, vol. 5, no. 2, pp. 202–225, Feb. 1996.

[22] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray, “Using vector quantization for image processing,” *Proc. IEEE*, vol. 81, no. 9, pp. 1326–1341, Sept. 1993.

[23] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: John Wiley & Sons, 1991.

[24] A. K. Dewdney and J. K. Vranch, “A convex partition of R^3 with applications to Crum’s problem and Knuth’s post-office problem,” *Utilitas Mathematica*, vol. 12, pp. 193–199, 1977.

[25] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*. Berlin, Germany: Springer-Verlag, 1987.

[26] P. Engel, “Geometric crystallography,” in *Handbook of Convex Geometry*, P. M. Gruber and J. M. Wills, Eds. Elsevier Science Publishers, pp. 989–1041, 1993.

[27] R. M. Fano, *Transmission of Information. A Statistical Theory of Communications*. M.I.T. Press and John Wiley & Sons, 1961.

[28] 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.

[29] A. Gersho, “On the structure of vector quantizers,” *IEEE Trans. Inform. Theory,* vol. IT-28, no. 2, pp. 157–166, Mar. 1982.

[30] A. Gersho and V. Cuperman, “Vector quantization: A pattern-matching technique for speech coding,” *IEEE Communications Magazine*, vol. 21, no. 9, pp. 15–21, Dec. 1983.

[31] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic Publishers, 1992.

[32] A. Gersho, S. Wang, and K. Zeger, “Vector quantization techniques in speech coding,” in *Advances in Speech Signal Processing,* S. Furui and M. M. Sondhi, Eds. New York, NY: Marcel Dekker, pp. 49–84, 1992.

[33] I. A. Gerson and M. A. Jasiuk, “Vector sum excited linear prediction (VSELP) speech coding at 8 kbps,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 461–464, Albuquerque, NM, Apr. 1990.

[34] I. A. Gerson and M. A. Jasiuk, “Vector sum excited linear prediction (VSELP),” in *Advances in Speech Coding*, B. S. Atal, V. Cuperman, and A. Gersho, Eds. Boston, MA: Kluwer Academic Publishers, pp. 69–79, 1991.

[35] J. D. Gibson and K. Sayood, “Lattice quantization,” in *Advances in Electronics and Electron Physics*, vol. 72, P. W. Hawkes, Ed. Boston, MA: Academic Press, pp. 259–330, 1988.

[36] R. M. Gray, “Vector quantization,” *IEEE ASSP Magazine*, vol. 1, no. 2, pp. 4–29, Apr. 1984.

[37] R. M. Gray, *Source Coding Theory*. Boston, MA: Kluwer Academic Publishers, 1990.

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

[39] P. M. Gruber and C. G. Lekkerkerker, *Geometry of Numbers*, 2nd ed. Amsterdam, the Netherlands: North-Holland, 1987.

[40] R. Hagen and P. Hedelin, “Robust vector quantization by a linear mapping of a block code,” submitted to *IEEE Trans. Inform. Theory,* Feb. 1995.

[41] D. Hartvigsen, “Recognizing Voronoi diagrams with linear programming,” *ORSA Journal of Computing*, vol. 4, no. 4, pp. 369–374, Fall 1992.

[42] S. Haykin, *Communication Systems*, 3rd ed. New York, NY: John Wiley & Sons, 1994.

[43] P. Hedelin, P. Knagenhjelm, and M. Skoglund, “Vector quantization for speech transmission,” in *Speech Coding and Synthesis*, W. B. Kleijn and K. K. Paliwal, Eds. Amsterdam, the Netherlands: Elsevier, pp. 311–345, 1995.

[44] T.-Y. Hwang, “Decoding linear block codes for minimizing word error rate,” *IEEE Trans. Inform. Theory*, vol. IT-25, no. 6, pp. 733–737, Nov. 1979.

[45] D. G. Jeong and J. D. Gibson, “Lattice vector quantization for image coding,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing,* vol. 3, pp. 1743–1746, Glasgow, Scotland, U.K., May 1989.

[46] D. G. Jeong and J. D. Gibson, “Uniform and piecewise uniform lattice vector quantization for memoryless Gaussian and Laplacian sources,” *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 786–804, May 1993.

[47] 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.

[48] V. Klee, “On the complexity of *d*-dimensional Voronoi diagrams,” *Archiv der Mathematik*, vol. 34, no. 1, pp. 75–80, Apr. 1980.

[49] P. Knagenhjelm, “Competitive learning in robust communication,” Ph.D. dissertation, Chalmers Univ. of Technology, Göteborg, Sweden, May 1993.

[50] S. Lin and D. J. Costello, Jr., *Error Control Coding. Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[51] K.-T. Lo and W.-K. Cham, “Subcodebook searching algorithm for efficient VQ encoding of images,” *IEE Proceedings-I,* vol. 140, no. 5, pp. 327–330, Oct. 1993.

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

[53] J. Makhoul, S. Roucos, and H. Gish, “Vector quantization in speech coding,” *Proc. IEEE,* vol. 73, no. 11, pp. 1551–1588, Nov. 1985.

[54] A. Méhes and K. Zeger, “Affine index assignments for binary lattice quantization with channel noise,” in *Proc. IEEE International Symposium on Information Theory*, p. 377, Whistler, British Columbia, Canada, Sept. 1995.

[55] A. Méhes and K. Zeger, “On the performance of affine index assignments for redundancy free source-channel coding,” in *Proc. Data Compression Conference*, p. 433, Salt Lake City, UT, Apr. 1995.

[56] A. Méhes and K. Zeger, “Binary lattice vector quantization with linear block codes and affine index assignments,” submitted to *IEEE Trans. Inform. Theory,* May 1996.

[57] D. Miller and K. Rose, “Combined source-channel vector quantization using deterministic annealing,” *IEEE Trans. Commun.*, vol. 42, nos. 2–4, pp. 347–356, Feb.–Apr. 1994.

[58] H. Minkowski, *Geometrie der Zahlen*.

[59] H. Minkowski, *Gesammelte Abhandlungen*. Leipzig, Germany: B. G. Teubner, 1911.

[60] N. Moayeri, “Fast vector quantization,” Ph.D. dissertation, The University of Michigan, Ann Arbor, MI, 1986.

[61] N. Moayeri and D. L. Neuhoff, “Time-memory treadeoffs in vector quantizer codebook searching based on decision trees,” *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 490–506, Oct. 1994.

[62] P. W. Moo and D. L. Neuhoff, “An asymptotic analysis of fixed-rate lattice vector quantization,” in *Proc. IEEE International Symposium on Information Theory and Its Applications*, vol. 1, pp. 409–412, Victoria, British Columbia, Canada, Sept. 1996.

[63] T. Moriya, “Two-channel conjugate vector quantizer for noisy channel speech coding,” *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 5, pp. 866–874, June 1992.

[64] S. Moshavi, “Multi-user detection for DS-CDMA communications,” *IEEE Communications Magazine*, vol. 34, no. 10, pp. 124–136, Oct. 1996.

[65] J. Møller, “Random Tessellations in $\mathbb{R}^d$,” *Advances in Applied Probability*, vol. 21, no. 1, pp. 37–73, Mar. 1989.

[66] N. M. Nasrabadi and R. A. King, “Image coding using vector quantization: A review,” *IEEE Trans. Commun.*, vol. 36, no. 8, pp. 957–971, Aug. 1988.

[67] D. L. Neuhoff, “On the performance of tree-structured vector quantization,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2277–2280, Toronto, Ontario, Canada, May 1991.

[68] T. Ohya, M. Iri, and K. Murota, “Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms,” *Journal of the Operations Research Society of Japan*, vol. 27, no. 4, pp. 306–337, Dec. 1984.

[69] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tessellations. Concepts and Applications of Voronoi Diagrams*. Chichester, England, U.K.: John Wiley & Sons, 1992.

[70] T. Ottosson, “Multirate schemes and multiuser decoding in DS/CDMA systems,” Tech. report 214L, School of Electrical and Computer Eng., Chalmers Univ. of Technology, Göteborg, Sweden, Nov. 1995.

[71] K. K. Paliwal and V. Ramasubramanian, “Vector quantization in speech coding: A review,” *Indian Journal of Technology,* vol. 24, no. 10, pp. 613–621, Oct. 1986.

[72] W. W. Peterson, *Error-Correcting Codes*. M.I.T. Press and John Wiley & Sons, 1961.

[73] J. G. Proakis, *Digital Communications*, 3rd ed. New York, NY: McGraw-Hill, 1995.

[74] M. Rabbani and P. W. Jones, *Digital Image Compression Techniques*. Bellingham, WA: SPIE Optical Engineering Press, 1991.

[75] V. Ramasubramanian, “Fast algorithms for nearest-neighbor serach and application to vector quantization,” Ph.D. dissertation, University of Bombay, Bombay, India, 1991.

[76] 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.

[77] D. J. Sakrison, “A geometric treatment of the source encoding of a Gaussian random variable,” *IEEE Trans. Inform. Theory*, vol. IT-14, no. 3, pp. 481–486, May 1968.

[78] E. G. Schukat-Talamazzini, M. Bielecki, H. Niemann, T. Kuhn, and S. Rieck, “A non-metrical space search algorithm for fast Gaussian vector quantization,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. II, pp. 688–691, Minneapolis, MN, Apr. 1993.

[79] E. Schulte, “Tilings,” in *Handbook of Convex Geometry*, P. M. Gruber and J. M. Wills, Eds. Elsevier Science Publishers, pp. 899–932, 1993.

[80] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, nos. 3 and 4, pp. 379–423 and 623–656, July and Oct. 1948.

[81] C. E. Shannon, “Coding theorems for a discrete source with a fidelity criterion,” in *IRE National Convention Record*, vol. 7, pt. 4, pp. 142–163, New York, NY, Mar. 1959.

[82] R. Sibson, “The Dirichlet tessellation as an aid in data analysis,” *Scandinavian Journal on Statistics: Theory and Applications*, vol. 7, no. 1, pp. 14–20, 1980.

[83] D. Slepian, “Permutation modulation,” *Proc. IEEE*, vol. 53, no. 3, pp. 228–236, Mar. 1965.

[84] P. F. Swaszek, “Vector quantization,” in *Communications and Networks. A Survey of Recent Advances*, I. F. Blake and H. V. Poor, Eds. New York, NY: Springer-Verlag, pp. 362–389, 1986.

[85] P. F. Swaszek, “Vector quantization based on block and spherical codes,” in *Proc. Conference on Information Sciences and Systems*, pp. 570–575, Baltimore, MD, Mar. 1991.

[86] P. F. Swaszek, “Syndrome-based VQ codebooks,” in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 14: Coding and Quantization*, R. Calderbank, G. D. Forney, Jr., and N. Moayeri, Eds. American Mathematical Society, pp. 75–81, 1993.

[87] P. F. Swaszek, “Vector quantization codebooks from the Nordstrom-Robinson code and Berlekamp’s negacyclic codes,” in *Proc. IEEE International Symposium on Information Theory*, p. 175, San Antonio, TX, Jan. 1993.

[88] H. L. Van Trees, *Detection, Estimation, and Modulation Theory. Part I: Detection, Estimation, and Linear Modulation Theory*. New York, NY: John Wiley & Sons, 1968.

[89] S. Vembu and S. Verdú, “The source-channel separation theorem revisited,” *IEEE Trans. Inform. Theory*, vol. 41, no. 1, pp. 44–54, Jan. 1995.

[90] E. Vidal, H. Rulot, F. Casacuberta, and J. Benedí, “Searching for nearest neighbours in constant average time with applications to discrete utterance speech recognition,” in *Proc. International Conference on Pattern Recognition*, pp. 808–810, Paris, France, Oct. 1986.

[91] E. Vidal, H. M. Rulot, F. Casacuberta, and J.-M. Benedí, “On the use of a metric-space search algorithm (AESA) for fast DTW-based recognition of isolated words,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 5, pp. 651–660, May 1988.

[92] A. J. Viterbi, *CDMA. Principles of Spread Spectrum Communication*. Reading, MA: Addison-Wesley, 1995.

[93] E. Viterbo and E. Biglieri, “Computing the Voronoi cell of a lattice: The diamond-cutting algorithm,” *IEEE Trans. Inform. Theory*, vol. 42, no. 1, pp. 161–171, Jan. 1996.

[94] G. Voronoï, “Nouvelles applications des paramètres continus à la théorie des formes quadratiques,” *Journal für die Reine und Angewandte Mathematik*, vol. 133, pp. 97–178, 1908; vol. 134, pp. 198–287, 1908; vol. 136, pp. 67–181, 1909.

[95] D. F. Watson, “Computing the *n*-dimensional Delaunay tessellation with application to Voronoi polytopes,” *The Computer J.*, vol. 24, no. 2, pp. 167–172, May 1981.

[96] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[97] K. Zeger and M. R. Kantorovitz, “Average number of facets per cell in tree-structured vector quantizer partitions,” *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 1053–1055, May 1993.

How to Evaluate Search Methods for Vector Quantization

Erik Agrell and Per Hedelin

Published in
Proceedings of the Nordic Signal Processing Symposium,
pp. 258–263, Ålesund, Norway, June 1994.

An error in the original version has been corrected as follows.

p. 263, col. 2, l. 1 from the bottom: 1989 1989 ➡ Sept. 1989

The layout has been revised.

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 $\left\{\mathbf{c}^{(i)}\right\}$ 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 **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 *m*th component of the transformed codevectors **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 $\left\{\mathbf{d}^{(i)}\right\}$ according to the bit-allocation. The triplets $\left\{\bar{d}_m^{(r)}, d_m^{-(r)}, d_m^{+(r)}\right\}$ are obtained as a natural part of this sorting procedure.

At run-time, a transformed source vector **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.

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

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, 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 $\mathbf{X}$. The maximum number of distance computations, over all $\mathbf{X}$, 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 code-

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.

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

book 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 $\rho = 0.75$.

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

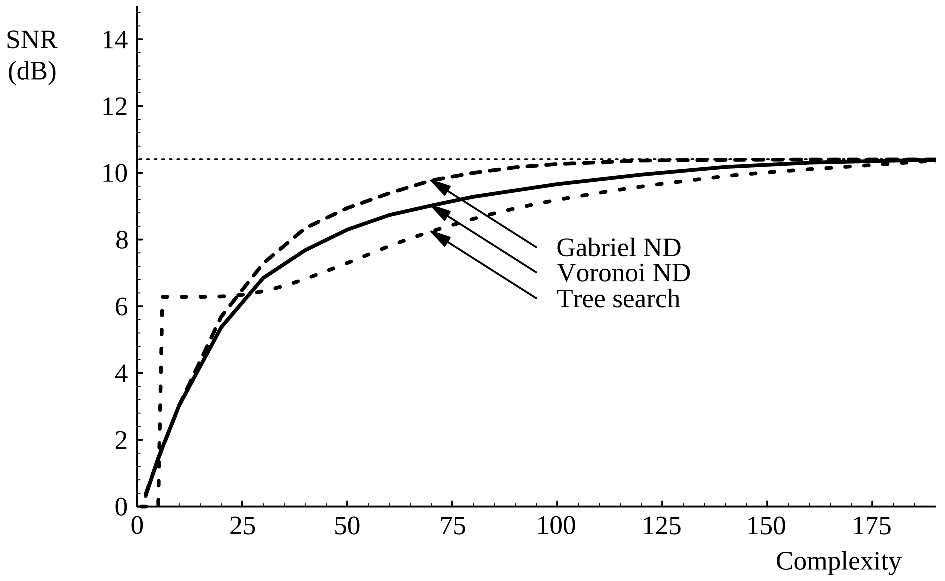


Figure 1. SNR as a function of complexity given by the number of distance measurements, for the vector quantizer with $\rho = 0$.

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.

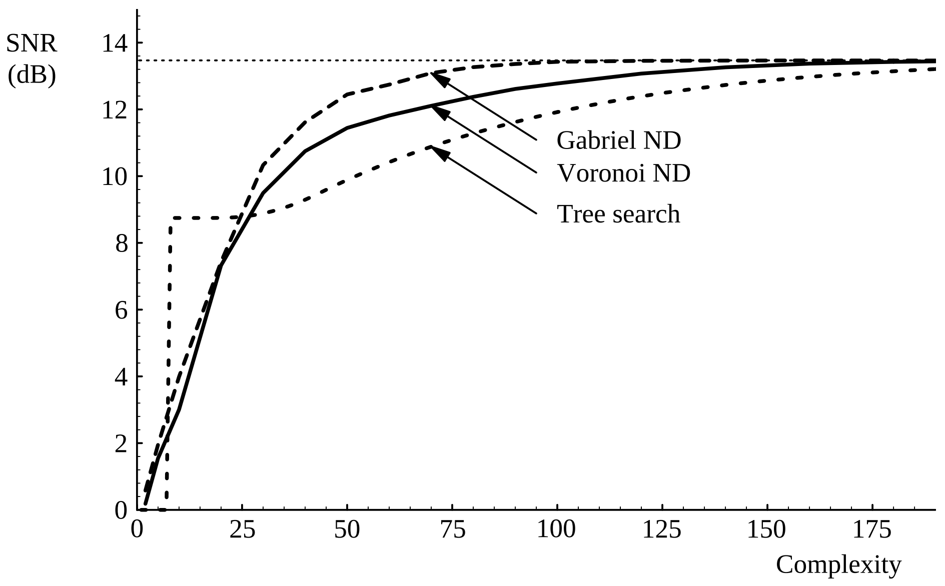


Figure 2. SNR as a function of complexity for $\rho = 0.75$.

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 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, Sept. 1989.

The Hadamard Transform—A Tool for Index Assignment

Petter Knagenhjelm and Erik Agrell

Published in
IEEE Transactions on Information Theory,
vol. 42, no. 4, pp. 1139–1151, July 1996.

Errors in the original version have been corrected as follows.

p. 1139, col. 1, l. 17 from the bottom: source-coding ➡ source coding

p. 1142, col. 2, l. 8 f.t.b.: $(l-q)$ ➡ $(1-q)$

Fig. 2(b): improved precision

p. 1146, col. 2, l. 3: reduced-echelon ➡ reduced echelon

p. 1147, col. 1, l. 17: $M-1-k$ and ➡ and $M-1-k$

Fig. 5, caption: $\rho = 0$ ➡ $\rho = 0.0$

p. 1150, col. 2, ll. 8–6 f.t.b.: indented one more step

p. 1151, col. 1, ll. 12 and 1 f.t.b.: *Communication* ➡ *Communications*

5 locations: index-assignment ➡ index assignment

2 locations: base 2 ➡ base-2

The layout has been revised.

Optimization of Lattices for Quantization

Erik Agrell and Thomas Eriksson

Submitted to
IEEE Transactions on Information Theory,
Dec. 1996.

Errors in the submitted version have been corrected as follows.

p. 30, l. 17: quantizer ➡ quantizers

p. 30, l. 2 from the bottom: Swazek ➡ Swaszek

The layout has been revised.

Lattice-Based Quantization, Part II

Thomas Eriksson and Erik Agrell

Published as
Technical Report 18,
Department of Information Theory,
Chalmers University of Technology,
Göteborg, Sweden,
Oct. 1996.

Part I of this work is included in article 3.

Errors in the original version have been corrected as follows.

- p. 9, ll. 11 and 9 from the bottom: a_T ➡ a_{T}
- p. 17, l. 1 f.t.b.: shows ➡ show
- p. 57, l. 2: Gray ➡ Gray, Jr.
- p. 57, l. 20: 259-3321988 ➡ 259–330, 1988
- p. 57, l. 8 f.t.b.: no. 3, pp. 379-423, 623-656, July-October 1948 ➡ nos. 3 and 4, pp. 379–423 and 623–656, July and October 1948
- p. 57, l. 4 f.t.b.: vol. 133, 134 and 1361908–1909 ➡ vol. 133, pp. 97–178, 1908; vol. 134, pp. 198–287, 1908; and vol. 136, pp. 67–181, 1909
- p. 58, l. 23: Eyuboglu and G. D. Forney ➡ Eyuboğlu and G. D. Forney, Jr.
- p. 58, l. 15 f.t.b.: Forney ➡ Forney, Jr.
- p. 58, l. 14 f.t.b.: 941-9481989 ➡ 941–958, Aug. 1989
- p. 58, l. 11 f.t.b.: *memory* ➡ *memory*,
- p. 58, l. 7 f.t.b. 1211-12261994 ➡ 1211–1227, 1994
- p. 59, l. 3: D. Y. Cheng ➡ D.-Y. Cheng
- p. 59, l. 6: Mitsumi ➡ Misumi
- p. 59, l. 2 f.t.b.: 5th ed. ➡ 5th ed.,

The layout has been revised.

Lattice-Based Quantization, Part II

Thomas Eriksson and Erik Agrell

Abstract In this report we study vector quantization based on lattices. A lattice is an infinite set of points in a regular structure. The regularity can be exploited in vector quantization to make fast nearest-neighbor search possible, and to reduce the storage requirements. Aspects of lattice vector quantization, such as scaling and truncation of the infinite lattice, are treated. Theory for high rate lattice quantization is developed, and the performance of lattice quantization of Gaussian variables is investigated. We also propose a method to exploit the lattice regularity to design fast search algorithms for unconstrained vector quantization. Experiments on Gaussian input data illustrate that the method performs well in comparison to other fast search algorithms.

1. Introduction

Vector quantization (VQ)[1] has since about 1980 become a popular technique for source coding of image and speech data. The popularity of VQ is motivated primarily by the theoretically optimal performance; no other source coding technique at equivalent delay can achieve better performance than optimal VQ. However, direct use of VQ suffers from a serious complexity barrier. Many authors have proposed constrained VQ structures to overcome the complexity, for example *multistage VQ* [1], *tree-structured VQ* [2–5], *vector-sum VQ* [6], *gain-shape VQ* [7], etc. Each of these solutions has disadvantages, in most cases a reduced performance. *Lattice VQ* [8, 9] is another constrained VQ technique, where the codevectors form a highly regular structure. The regular structure makes compact storage and fast *nearest-neighbor search* (finding the closest codevector to an input vector) possible, but also leads to performance loss.

Another line of research, also aimed to overcome the complexity barrier of VQ, is design of fast search methods for unconstrained quantizers. Due to the presumed lack

[1] With VQ, we will sometimes mean *vector quantization*, and sometimes *vector quantizer*, with the distinction left to the context.

of structure in such quantizers[2], nearest-neighbor search for unconstrained VQ is considerably more difficult than search of a constrained VQ. Algorithms for fast nearest-neighbor search of unconstrained VQ include for example *neighbor descent* methods [10, 11], where the complexity of a full search is avoided by precomputing an *adjacency table*, consisting of all neighbors to all VQ points. Other methods are the *anchor point* algorithm [12], where codevectors are excluded from the search by the triangle inequality, and the *K-d tree* technique [13], where a prestored tree structure helps in avoiding unnecessary operations.

In this report, we discuss *lattice-based quantization*[3] as a solution of the complexity problem. Lattice-based quantization is a generalization of conventional lattice quantization, by allowing modifications of the regular lattice structure while still maintaining a local lattice-similarity. In the first part of the report, conventional lattice quantization is treated. After the introduction and VQ preliminaries in chapter 1 and 2, we present high rate theory for lattice VQ for Gaussian variables in chapter 3. The high rate theory leads to design rules for lattice VQ, and formulas for asymptotic performance. Further, the performance of lattice VQ for a Gaussian input pdf is compared to the performance of pdf-optimized VQ. An important task in lattice VQ design is the *truncation* of an infinite-size lattice, to include the desired number of codevectors in the VQ. Other important aspects are for example the choice of lattice, and scaling of the source, to get a good performance. These aspects are treated from a practical perspective in chapter 3, and solutions are found, based on the lattice high rate theory. In many previous reports, the focus has been on high-dimensional lattice quantization, due to the *asymptotic equipartition property* (AEP); when the dimension grows to infinity, the d-dimensional probability density of a memoryless input source becomes more and more localized to a ”typical” region, inside which the density is approximately uniform [15]. Thus, a lattice quantizer, with an inherent uniform distribution of codevectors, can be expected to work well for high dimensions. We have instead focused on low-dimensional (2–5 dimensions) lattice VQ, since several interesting areas in speech and image coding employ low-dimensional parameter vectors.

The density of the codevectors in a lattice quantizer is uniform, which may inflict on the efficiency of lattice quantization for nonuniform sources. We propose a novel VQ design concept in chapter 4, with the goal to combine some of the desirable properties of a lattice VQ with the good performance of a pdf-optimized VQ. The VQ is initialized with a truncated lattice, and an adjacency table for the lattice is computed.

[2] A pdf-optimized unconstrained VQ is generally far from unstructured, but the structure may be difficult to find and exploit.

[3] Most of the conclusions in this report holds for *tessellation quantizers* as well. More about tessellations can be found in [14].

Then, during the training, the quantizer is updated to keep the neighbors as given by the lattice adjacency table. By example, we show that this *lattice attraction* can be imposed with almost no performance loss at all for a Gaussian input pdf. A neighbor descent algorithm [11], modified to suit the special requirements of the lattice-attracted quantizers, is presented in chapter 5. The performance of the new neighbor descent method is reported in chapter 6, together with the performance of direct lattice quantization of Gaussian variables. Finally, a summary is given in chapter 7.

2. VECTOR QUANTIZATION

In this chapter, we present vector quantization theory. Necessary optimality conditions for a VQ is given, and theory for high rate quantization is discussed.

2.1 Definitions

A VQ $\mathcal{Q}$ of size N and dimension d is a mapping from a vector in the d-dimensional Euclidean space $\mathbb{R}^d$ into a finite reproduction set $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}$:

$$\mathcal{Q} : \mathbb{R}^d \to \mathcal{C} . \tag{2.1}$$

The set $\mathcal{C}$, denoted the *codebook*, contains N *codevectors* $\mathbf{c}_k, k = 1,2,...,N$, each a vector in $\mathbb{R}^d$. The index k of the codevectors is denoted *codeword*. The rate R of the quantizer is defined as $\log_2(N)/d$ [bits per sample]. The definition of $\mathcal{Q}$ in (2.1) partitions $\mathbb{R}^d$ into N disjoint regions, each with a corresponding codevector $\mathbf{c}_k$.

The vector quantizer can be decomposed in two components, the encoder and the decoder. The encoder $\mathcal{E}$ maps from $\mathbb{R}^d$ to the index set $I = \{1, 2, ..., N\}$

$$\mathcal{E} : \mathbb{R}^d \to I , \tag{2.2}$$

and the decoder $\mathcal{D}$ maps the index set into the reproduction set $\mathcal{C}$, i.e.,

$$\mathcal{D} : I \to \mathbb{R}^d . \tag{2.3}$$

With this notation, the quantization operation can be written as a cascade of the encoder and decoder:

$$\mathcal{Q}(\mathbf{x}) = \mathcal{D}(\mathcal{E}(\mathbf{x})). \tag{2.4}$$

In this report, we will measure the performance by the statistical mean of the squared Euclidean distance measure,

$$D = \mathrm{E}\left[\|\mathbf{x} - \mathcal{Q}(\mathbf{x})\|^2\right]. \tag{2.5}$$

The mean squared error criterion is only one of many possible distortion measures, but it has the advantage of being widely used and is mathematically simple.

2.2 Optimality conditions

In VQ design, the aim is to find encoder and decoder rules to minimize the chosen distortion measure. For the squared Euclidean distance measure (2.5) (with a decoder $\mathcal{D}(i) = \mathbf{c}_i$), it can be readily shown [16] that for a fixed partition Ω_k of the input space, the codevectors $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}$ should be chosen as the centroid of the vectors in the region,

$$\mathbf{c}_k = \mathrm{E}[\mathbf{x} | \mathbf{x} \in \Omega_k] \tag{2.6}$$

to minimize the expected distortion. (2.6) is often called *the centroid condition*. If instead the set of codevectors is fixed, the partition should be the *nearest neighbor partition*:

$$\Omega(\mathbf{c}_k) = \Omega_k = \left\{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{c}_k\|^2 \le \|\mathbf{x} - \mathbf{c}_i\|^2 \text{ for all } i \in I \right\} \tag{2.7}$$

with the corresponding encoder rule

$$\mathcal{E}(\mathbf{x}) = \underset{\mathrm{i} \in I}{\mathrm{argmin}} \|\mathbf{x} - \mathbf{c}_i\|^2, \tag{2.8}$$

together with rules to solve ties. The regions Ω_k are often referred to as *Voronoi regions,* after the author of [17].

We see that both the encoder and the decoder are completely specified by the codebook $\mathcal{C}$, so finding optimal encoder and decoder rules is equivalent to finding the optimum set of codevectors $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}$.

The centroid condition (2.6) and the nearest neighbor partition (2.7) are necessary but not sufficient for a VQ to be optimal in the mean square sense. Sufficient conditions for a globally optimal VQ have never been presented (except for some special cases), and a quantizer fulfilling the necessary conditions may be far from optimal. This makes VQ design a delicate problem.

Using the nearest neighbor condition, the *Voronoi neighbors* to a Voronoi region Ω_k in a VQ can be defined as

$$\mathcal{A}_k = \{ i \in [1, N] : \Omega_i \cap \Omega_k \ne \varnothing \} \tag{2.9}$$

that is, the set of codevectors whose Voronoi regions share a face with Ω_k. With this definition, the nearest neighbor partition can be reformulated as

$$\Omega_k = \left\{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{c}_k\|^2 \le \|\mathbf{x} - \mathbf{c}_i\|^2 \text{ for all } i \in \mathcal{A}_k \right\}, \tag{2.10}$$

which illustrates that the Voronoi region is defined by a subset of the inequalities in (2.7). The new definition of the nearest neighbor partition shows that to find the optimum codevector to a given input vector $\mathbf{x}$, it suffices to find a codevector whose Voronoi neighbors all have greater distance to the input vector. This can be exploited in fast search algorithms, as described in chapter 5.

2.3 High rate theory

In [18] and [16], it is shown that for high resolution VQs, the optimal reconstruction point density $\lambda(\mathbf{x})$ for quantization of a stochastic vector process $\mathbf{x}$ with pdf $f_{\mathbf{x}}(\mathbf{x})$ is given by

$$\lambda(\mathbf{x}) = a \cdot f_{\mathbf{x}}^{d/(d+2)}(\mathbf{x}) \tag{2.11}$$

where d is the dimension of the VQ, and a is a normalizing constant. For a quantizer with the above optimal point density, we have for high rates [16]

$$D \geq \frac{d \cdot \Gamma^{2/d}(d/2+1)}{(d+2)\cdot \pi} \left(\int f_{\mathbf{x}}(\mathbf{x})^{d/(d+2)} \right)^{(d+2)/d} \cdot 2^{-2R}, \tag{2.12}$$

where R is the rate of the quantizer, in bits per dimension.

For an uncorrelated Gaussian pdf, the above expression can be simplified to the Gaussian lower bound (GLB)

$$D_{\mathrm{GLB}} \geq 2^{-2R} \cdot f(d) \cdot \sigma_{\mathbf{x}}^2, \tag{2.13}$$

where

$$f(d) = \frac{2}{d}\left(\frac{d+2}{d}\right)^{d/2} \Gamma^{2/d}(d/2+1), \tag{2.14}$$

and

$$\sigma_{\mathbf{x}}^2 = \mathrm{E}\left[\left\|\mathbf{x}-\mathbf{m}_{\mathbf{x}}\right\|^2\right] = \int \left\|\mathbf{x}-\mathbf{m}_{\mathbf{x}}\right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \tag{2.15}$$

$$\mathbf{m}_{\mathbf{x}} = \mathrm{E}[\mathbf{x}] = \int \mathbf{x} \cdot f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}. \tag{2.16}$$

Knagenhjelm [19] shows experimentally that the Gaussian lower bound is not only a lower bound, but also a good approximation to the actual performance of a well-trained vector quantizer, if the rate is high.

3. LATTICE QUANTIZATION

In this chapter, we will treat lattice quantization, both from a theoretical and a practical perspective. High rate theory for lattice quantization of iid Gaussian variables is derived, leading to formulas for lattice VQ design and performance. Practical issues in lattice VQ design, such as truncation and scaling of the lattice, are also treated.

3.1 Definitions

A *lattice* is an infinite set of points, defined as

$$\Lambda = \left\{ \mathbf{B}^T \cdot \mathbf{u} : \quad \mathbf{u} \in \mathbb{Z}^d \right\} \tag{3.1}$$

where $\mathbf{B}$ is the *generator matrix* of the lattice. The rows of $\mathbf{B}$ constitute a set of d linearly independent *basis vectors* for the lattice,

$$\mathbf{B} = \left[\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_d\right]^T \tag{3.2}$$

Thus, the lattice Λ consists of all linear combinations of the basis vectors, with integer coefficients.

The *theta function* of the lattice gives the number of lattice points $\mathbf{c}_i$ at a specific distance from the origin, i.e. points within a *shell*. The theta function for many standard lattices can be found in [9].

The *fundamental parallelotope* of the lattice is defined as the parallelotope

$$z_1\mathbf{b}_1 + \ldots + z_d\mathbf{b}_d \qquad (0 \le z_i < 1). \tag{3.3}$$

Associated with each lattice point is a Voronoi region. Due to the regular structure of lattices, all Voronoi regions in a lattice are simply translations of the Voronoi region $\Omega(\mathbf{0})$ around the zero lattice point. $\Omega(\mathbf{0})$ is referred to as the *lattice Voronoi region* Ω, with the definition

$$\Omega = \left\{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|^2 \le \|\mathbf{x} - \mathbf{c}\|^2 \text{ for all } \mathbf{c} \in \Lambda\right\} \tag{3.4}$$

The *normalized second moment* of a Voronoi region $\Omega(\mathbf{c}_i)$ is defined to be

$$G = \frac{1}{d}\left[\mathrm{vol}\left(\Omega(\mathbf{c}_i)\right)\right]^{-1-2/d} \int\limits_{\Omega(\mathbf{c}_i)} \|\mathbf{x} - \mathbf{c}_i\|^2 d\mathbf{x}, \tag{3.5}$$

where $\mathrm{vol}\left(\Omega(\mathbf{c}_i)\right)$ is the volume of the Voronoi region around $\mathbf{c}_i$. Since $\Omega(\mathbf{c}_i)$ is a translation of Ω, $\Omega(\mathbf{c}_i) = \Omega + \mathbf{c}_i$, we can write

$$G = \frac{1}{d}\left[\mathrm{vol}(\Omega)\right]^{-1-2/d} \int\limits_{\Omega} \|\mathbf{x}\|^2 d\mathbf{x}, \tag{3.6}$$

which illustrates that G is independent of i. The constant G is from now on be referred to as the *quantization constant* of the lattice, since it describes the mean squared error per dimension for quantization of an infinite uniform distribution, if the volume of the Voronoi region is normalized to one.

Lattice quantization is a special class of vector quantization, with the codebook having a highly regular structure. Any codevector $\mathbf{c}_k \in \mathcal{C}$ in a lattice quantizer can be written on the form

$$\mathbf{c}_k = \mathbf{B}^T \cdot \mathbf{u}_k \tag{3.7}$$

where $\mathbf{u}_k$ is one of N given integer vectors, and $\mathbf{B}$ is the generator matrix of the lattice. Alternatively, a lattice VQ can be described as the intersection between a *lattice* Λ and a *shape* $\mathcal{S}$,

$$\mathcal{C} = \Lambda \cap \mathcal{S} \tag{3.8}$$

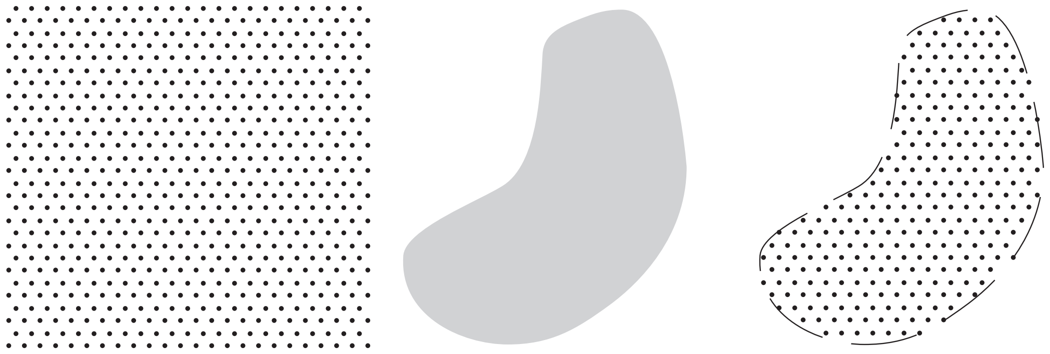


Figure 3.1. Illustration of lattice truncation. Left: a lattice Λ, Center: a shape $\mathcal{S}$, Right: the resulting lattice quantizer $\mathcal{C}$.

where $\mathcal{S}$ is a d-dimensional bounded region in $\mathbb{R}^d$. An example is shown in figure 3.1.

The design of a lattice VQ can now be separated into finding a good lattice, specified through its generator matrix **B**, and a good shape $\mathcal{S}$. In addition, a scale factor for the lattice must be found, and an assignment of indices to the codevectors. These problems will be treated in the following sections.

Applications of lattice vector quantization include, e.g., image coding [20, 21] and speech coding [22, 23]. Moayeri et al. superimposed a fine lattice upon a source-optimized unstructured VQ to achieve a fast two-step search method [24, 25]. Kuhlmann and Bucklew [26], Swaszek [27] and Eriksson [28] connects lattices with different scaling into one “piecewise uniform” codebook, to approximate nonuniform source pdfs. In [14], an overview of applications including lattice VQ is presented.

3.2 Theory for high rate lattice quantization

In this section, we derive expressions for the distortion of lattice quantization of iid Gaussian vectors, when the rate R of the quantizer tends to infinity. Eyuboğlu and Forney [29], and Jeong and Gibson [30], have previously worked with high rate theory for lattice quantization, but to the authors’ knowledge, simple analytical expressions for the optimal truncation and performance of d-dimensional lattice quantizers has not been presented before. A major difference between the high rate lattice theory presented here and the usual high rate theory for optimal quantization (section 2.3), is that for lattice quantization, it is necessary to explicitly consider overload distortion, while the usual high rate theory only permits granular distortion.

We assume an iid Gaussian input pdf, with zero mean, unit variance samples. However, in the end of this section we discuss a generalization of the results.

After some definitions, two theorems concerning the distortion of a lattice VQ as a function of the rate and truncation are given. The optimal truncation radius, and the corresponding distortion, are found by setting the derivative of the distortion to zero.

A *d-sphere* is a d-dimensional sphere, defined as

$$S_d(a) = \left\{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \le a \right\}. \tag{3.9}$$

We assume a truncation shape in the form of a d-sphere with radius a_T (figure 3.2), so that

$$\mathcal{C} = (\Lambda - \mathbf{v}) \cap S_d(a_\mathrm{T}), \tag{3.10}$$

where $\mathbf{v}$ is an arbitrary vector (see the discussion in section 3.4, and (3.33)).

We subdivide the d-dimensional space into two (nonspherical) subregions: a *granular* region $\mathcal{G}$, which we define as the union of lattice Voronoi regions around all codevectors,

$$\mathcal{G} = \bigcup_{\mathbf{c}_i \in \mathcal{C}} (\Omega + \mathbf{c}_i), \tag{3.11}$$

and an *overload* region $\overline{\mathcal{G}}$, which is the rest of the space, so that $\mathcal{G} \cup \overline{\mathcal{G}} = \mathbb{R}^d$ and $\mathcal{G} \cap \overline{\mathcal{G}} = \varnothing$. Figure 3.2 illustrates the granular and overload regions for a two-dimensional lattice VQ, based on the well-known hexagonal lattice A_2.

The total distortion D of the lattice quantizer can be separated into a granular component, $D_\mathcal{G}$, and an overload component, $D_{\overline{\mathcal{G}}}$,

$$D = \int_{\mathbb{R}^d} \left\|\mathbf{x} - \mathbf{c}^*\right\|^2 f_\mathbf{x}(\mathbf{x}) d\mathbf{x} = \int_{\mathcal{G}} \left\|\mathbf{x} - \mathbf{c}^*\right\|^2 f_\mathbf{x}(\mathbf{x}) d\mathbf{x} + \int_{\overline{\mathcal{G}}} \left\|\mathbf{x} - \mathbf{c}^*\right\|^2 f_\mathbf{x}(\mathbf{x}) d\mathbf{x} = D_\mathcal{G} + D_{\overline{\mathcal{G}}}, \tag{3.12}$$

where $\mathbf{c}^*$ denotes the codevector in the codebook $\mathcal{C}$ that is closest to the input vector $\mathbf{x}$. We now give two theorems, leading to simple approximations of the granular and the overload distortion of lattice quantization. In the first theorem, we write the overload distortion as the distortion given a high codevector density close to the surface of the truncation sphere, plus an error term. The second theorem is mainly based on the smoothness of the Gaussian pdf, so that the pdf within the granular Voronoi regions is nearly uniform, if the Voronoi regions are small. Both theorems are proved in appendix A.

Theorem I: The overload distortion is given by

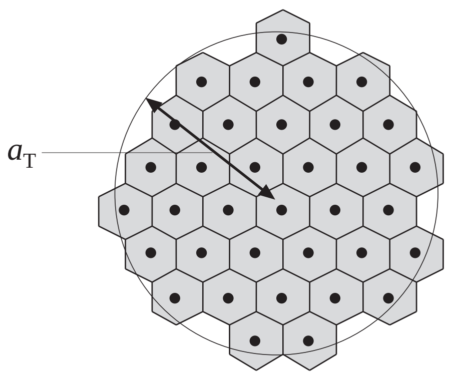


Figure 3.2. Illustration of the granular region (the gray area) and the overload region (everything but the gray area) of a 2-dimensional lattice quantizer.

$$D_{\bar{\mathcal{G}}} = f_{\bar{\mathcal{G}}}(d) \cdot a_{\mathrm{T}}^{d-4} \cdot e^{-a_{\mathrm{T}}^2/2} \cdot \left(1 + \varepsilon_{\bar{\mathcal{G}}}\right) \tag{3.13}$$

where $f_{\bar{\mathcal{G}}}(d) = \left(2^{d/2-2} \cdot \Gamma(d/2)\right)^{-1}$. For asymptotically high rates *R*, and the truncation radius a_{T} suitably chosen, $\varepsilon_{\bar{\mathcal{G}}}$ tends to zero.

Theorem II: The granular distortion is given by

$$D_{\mathcal{G}} = f_{\mathcal{G}}(d) \cdot a_{\mathrm{T}}^2 \cdot 2^{-2R} \cdot \left(1 + \varepsilon_{\mathcal{G}}\right) \tag{3.14}$$

where $f_{\mathcal{G}}(d) = G \cdot d \cdot \pi \cdot \Gamma(d/2+1)^{-2/d}$. For asymptotically high rates *R*, and the truncation radius a_{T} suitably chosen, $\varepsilon_{\mathcal{G}}$ tends to zero.

The total distortion, *D*, can be written

$$D = D_{\mathcal{G}} + D_{\bar{\mathcal{G}}} = \left(f_{\mathcal{G}}(d) \cdot a_{\mathrm{T}}^2 \cdot 2^{-2R} + f_{\bar{\mathcal{G}}}(d) \cdot a_{\mathrm{T}}^{d-4} \cdot e^{-a_{\mathrm{T}}^2/2}\right) \cdot (1+\varepsilon), \tag{3.15}$$

where the error term ε tends to zero when *R* grows towards infinity. For the moment, we exclude the error term, and seek the minimum of

$$\hat{D} = \hat{D}_{\mathcal{G}} + \hat{D}_{\bar{\mathcal{G}}} = f_{\mathcal{G}}(d) \cdot a_{\mathrm{T}}^2 \cdot 2^{-2R} + f_{\bar{\mathcal{G}}}(d) \cdot a_{\mathrm{T}}^{d-4} \cdot e^{-a_{\mathrm{T}}^2/2}. \tag{3.16}$$

In appendix A.4, it is shown that the minimum value of $\hat{D}$ is also the minimum value of *D*. To find the value of the truncation radius a_{T} that minimizes the distortion, we differentiate $\hat{D}$ with respect to a_{T}:

$$\frac{\partial \hat{D}}{\partial a_{\mathrm{T}}} = 2 \cdot f_{\mathcal{G}}(d) \cdot a_{\mathrm{T}} \cdot 2^{-2R} + f_{\bar{\mathcal{G}}}(d) \cdot (d-4) \cdot a_{\mathrm{T}}^{d-5} \cdot e^{-a_{\mathrm{T}}^2/2} - f_{\bar{\mathcal{G}}}(d) \cdot a_{\mathrm{T}}^{d-3} \cdot e^{-a_{\mathrm{T}}^2/2}. \tag{3.17}$$

Since $\hat{D}$ is a convex and continuous function in the interesting region (see section A.4), we get the condition for minimal distortion by setting the derivative to zero,

$$\frac{\partial \hat{D}}{\partial a_{\mathrm{T}}} = 0 \quad \Leftrightarrow \quad f_{\bar{\mathcal{G}}}(d) \cdot a_{\mathrm{T,opt}}^{d-6} \cdot e^{-a_{\mathrm{T,opt}}^2/2} \cdot \left(a_{\mathrm{T,opt}}^2 + 4 - d\right) = 2 \cdot f_{\mathcal{G}}(d) \cdot 2^{-2R}. \tag{3.18}$$

where $a_{\mathrm{T,opt}}$ is the value of a_{T} that minimizes the distortion. We observe that by multiplying both sides of (3.18) with a_{T}^2, we get

$$\hat{D}_{\bar{\mathcal{G}}} \cdot \left(a_{\mathrm{T,opt}}^2 + 4 - d\right) = 2\hat{D}_{\mathcal{G}} \tag{3.19}$$

where $\hat{D}_{\bar{\mathcal{G}}}$ and $\hat{D}_{\mathcal{G}}$ are given by (3.16). We get

$$\frac{\hat{D}_{\bar{\mathcal{G}}}}{\hat{D}_{\mathcal{G}}} = \frac{2}{a_{\mathrm{T,opt}}^2 + 4 - d}. \tag{3.20}$$

In appendix A it is shown that $a_{\mathrm{T,opt}}$ tends to infinity when *R* approaches infinity. We conclude that the total distortion is dominated by the granular distortion, when the rate tends to infinity,

$$\frac{D_{\bar{\mathcal{G}}}}{D_{\mathcal{G}}} \to 0 \text{ when } R \to \infty. \tag{3.21}$$

Returning to (3.18), and taking the logarithm of both sides, we have

$$-\frac{a_{\mathrm{T,opt}}^2}{2} + (d-6)\cdot\ln\left(a_{\mathrm{T,opt}}\right) + \ln\left(a_{\mathrm{T,opt}}^2 + 4 - d\right) = -R\cdot 2\ln 2 + \ln\left(\frac{2\cdot f_{\mathcal{G}}(d)}{f_{\bar{\mathcal{G}}}(d)}\right), \quad (3.22)$$

or, equivalently,

$$a_{\mathrm{T,opt}}^2 - (d-4)\cdot\ln\left(a_{\mathrm{T,opt}}^2\right) - 2\cdot\ln\left(1 + \frac{4-d}{a_{\mathrm{T,opt}}^2}\right) = 4\ln 2\cdot R - 2\cdot\ln\left(\frac{2\cdot f_{\mathcal{G}}(d)}{f_{\bar{\mathcal{G}}}(d)}\right). \quad (3.23)$$

Since $a_{\mathrm{T,opt}}$ tends to infinity for rates approaching infinity, both sides are dominated by their first terms, resulting in

$$a_{\mathrm{T,opt}}^2 \approx R\cdot 4\ln 2 \quad \text{when } R \to \infty. \quad (3.24)$$

that is, the optimal truncation radius $a_{\mathrm{T,opt}}$ is proportional to the square root of R for asymptotically high rates.

The total distortion (3.15) can now be written

$$D = g(R,d)\cdot 2^{-2R}, \quad (3.25)$$

where $g(R,d)$ is approximated using (3.21) and (3.24),

$$g(R,d) \approx 4\cdot\ln 2\cdot f_{\mathcal{G}}(d)\cdot R \quad \text{when } R \to \infty. \quad (3.26)$$

It is easy to generalize the formulas to arbitrary variance, by making the substitution $\mathbf{y} = \mathbf{x}\cdot\sqrt{\sigma_{\mathbf{y}}^2/d}$ (see (3.27)–(3.29)). If we compare the lattice VQ distortion with the distortion of a pdf-optimized quantizer (2.13), we see that the discrepancy increases with the rate. This can be observed in figure 3.7, section 3.6, where optimal VQ and lattice VQ are compared.

(3.25) is only proven for rates approaching infinity, but we have experimentally verified that the formulas also hold for realistic rates. In figure 3.3, the experimental performance of lattice quantization (see table 6.1) is compared to the high rate theory

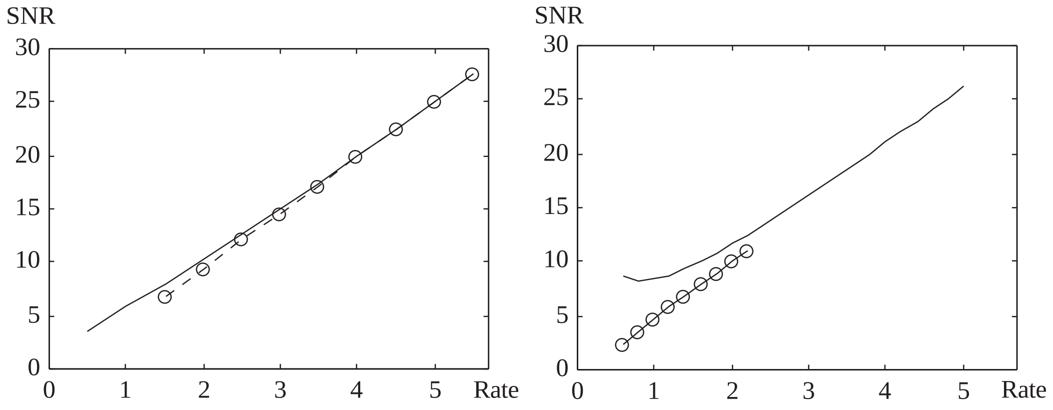


Figure 3.3. Experimental performance for lattice quantization of an iid Gaussian pdf (circles), and performance predicted by lattice VQ high rate theory (line). Left: 2 dimensions. Right: 5 dimensions.

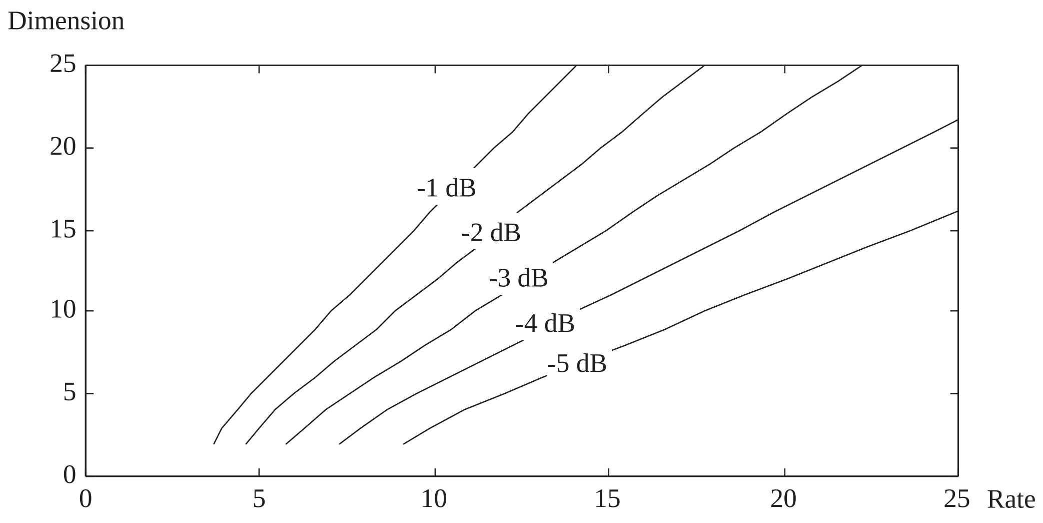


Figure 3.4. Estimated performance loss for a lattice VQ compared with a pdf-optimized VQ. The curves indicate rate and dimension for lattice quantizers with performance loss from 1 to 5 dB.

results, for quantization of 2- and 5-dimensional Gaussian variables.

With this theoretical derivation of lattice VQ performance, we have two asymptotical lattice VQ results: the asymptotic equipartition property predicts that a lattice VQ performs better for high dimensions, while the high rate theory predicts that a lattice VQ performs worse for high rates. These results are illustrated in figure 3.4, where each curve indicates a specific performance loss compared to a pdf-optimized VQ. The curves in figure 3.4 were computed by use of the high rate lattice theory (3.25) and the Gaussian high rate lower bound in (2.13).

The formulas above were derived for iid Gaussian densities, with zero mean, unit variance samples, but it is straightforward to generalize the theory to arbitrary variance and mean. The conclusions should be similar also for correlated Gaussian data, but the theory is more complicated for correlated variables. By simple modifications, the formulas can be used for a generalized Gaussian pdf. Some of the results may also be possible to generalize to other pdfs. For all unbounded pdfs, such as Gaussian, Laplace, Gamma, etc., the size of the granular region must increase when the rate increases, for the overload distortion to be zero for an infinite rate. Thus, the granular region includes parts of the space with lower and lower pdf. Therefore, the larger the rate, the more the point density of an optimal quantizer, given by (2.11), differ from the uniform point density of a lattice quantizer. Based on the above reasoning, and on our experience of high rate theory for Gaussian pdfs, we believe that the suboptimality of lattice quantizers for high rates holds under far more general conditions than for iid Gaussian distributions.

Substituting as discussed above, to get formulas that are valid for arbitrary input signal variance, we conclude the high rate lattice theory in the following three points:

- The optimal squared truncation radius is proportional to the rate for high rates,

$$a_{\mathrm{T,opt}}^2 \approx R \cdot \frac{4 \ln 2}{d} \cdot \sigma_{\mathbf{y}}^2 \;\text{ when }\; R \to \infty . \tag{3.27}$$

- For high rates, the granular distortion dominates over the overload distortion,

$$\frac{D_{\overline{\mathcal{G}}}}{D_{\mathcal{G}}} \to 0 \;\text{ when }\; R \to \infty . \tag{3.28}$$

- For high rates, the performance of lattice quantizers, as given by the high rate formula

$$D \approx R \cdot 2^{-2R} \cdot G \cdot 4 \cdot \ln 2 \cdot \pi \cdot \Gamma(d/2+1)^{-2/d} \cdot \sigma_{\mathbf{y}}^2 \;\text{ when }\; R \to \infty , \tag{3.29}$$

  is inferior to the performance of optimal vector quantizers, given by the Gaussian lower bound (2.13).

3.3 Selection of lattice

The choice of lattice is of course of major importance for the performance of a lattice VQ. Ideally, the lattice should be selected to suit both the actual pdf and the truncation. However, for high rate quantization of smooth pdfs, the choice of lattice is fairly independent of input pdf and truncation [16]. For these cases, the lattice can be chosen based on its quantization performance for an infinite uniform pdf. This choice is motivated by high rate theory; for high rates, the pdf in each Voronoi region can be expected to be approximately uniform, at least for reasonably smooth pdfs (such as the Gaussian pdf). Further, the performance of infinite uniform lattice quantization, given by the quantization constant G, is easily found in the literature for many lattices.

Conway and Sloane [9] give values of the quantization constant G and lattice basis **B** for several lattices. For example, the best known lattices for quantization of infinite uniform pdfs in 2 and 5 dimensions are generated by, respectively,

$$\mathbf{B} = s \begin{bmatrix} 2 & 0 \\ 1 & \sqrt{3} \end{bmatrix} \tag{3.30}$$

and

$$\mathbf{B} = s \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.31}$$

where s is a scale factor to be determined[4]. The first is the well-known hexagonal grid (figure 3.2), also denoted the A_2 lattice, and the second is the D_5^* lattice. The best known lattices for quantization of infinite uniform pdfs in 2–5 dimensions are A_2, D_3^*, D_4^* and D_5^*, respectively. These lattices are employed in our experiments in chapter 6. In [14], lattices for quantization purposes are thoroughly studied.

3.4 Truncation and scaling

As described previously in this chapter, a lattice quantizer is the intersection between a lattice Λ and a shape $\mathcal{S}$. The procedure to reject lattice points outside the shape, called *truncation* of the lattice, is of major importance for the performance of the resulting lattice quantizer.

Truncation for known distributions: Jeong and Gibson [30] argue that in a good lattice VQ, the lattice should be truncated by a contour of constant probability density for the considered source, and design lattice VQs for Gaussian and Laplacian data. For the Laplacian pdf, this leads to truncation by a d-octahedron, which, mostly in combination with the integer lattice $\mathbb{Z}^d$, has received much attention since Fischer introduced the structure (Pyramid VQ) in the mid-80's. A recent reference on this topic is [31]; see also Swazek [32]. For a Gaussian pdf, the iso-probability contours are ellipsoids, and a corresponding truncating shape $\mathcal{S}$ is described by

$$\mathcal{S} = \left\{ \mathbf{x} \in \mathbb{R}^d : \mathbf{x}^{\mathrm{T}} \mathbf{C}_{\mathbf{x}}^{-1} \mathbf{x} < a^2 \right\} \tag{3.32}$$

where $\mathbf{C}_{\mathbf{x}}$ is the covariance matrix of the Gaussian input distribution, and a is a constant, determining the size of the ellipsoid. To truncate a lattice to the correct number of VQ points, the radius a above must be determined. An approximate value of a can be found by using the volume of the lattice Voronoi region, and for certain rates, a can be found by use of the theta function of the lattice.

A problem that may occur when lattices are truncated to a desired number of points is that a lattice normally has many points lying on the same distance from the origin (shell), and the truncation procedure may be required choose a few among those. To prevent lattice points to fall on the boundary, an arbitrary vector $\mathbf{v} \in \mathbb{R}^d$ can be added to the shape prior to the truncation:

$$C = \Lambda \cap (\mathcal{S} + \mathbf{v}) \text{ (or, equivalently, } C = (\Lambda - \mathbf{v}) \cap \mathcal{S} \text{)}. \tag{3.33}$$

After the truncation, the truncated lattice is moved to make the mean of all codevectors equal to the mean of the source. The choice of $\mathbf{v}$ can affect the performance of the resulting quantizer. We have experimented with four different methods to select $\mathbf{v}$:

I $\mathbf{v}$ is set to zero.

[4] Lattices can of course also be rotated and translated, but for high rates and smooth pdfs, these operations have little influence of the performance of a lattice VQ.

II $\mathbf{v}$ is selected as a very small (small compared to the basis vectors of the lattice) stochastic vector.

III $\mathbf{v}$ is selected as a stochastic vector with length in parity with the basis vectors of the lattice.

IV $\mathbf{v}$ is selected to minimize the energy of the resulting quantizer C,

$$\mathbf{v} = \underset{\mathbf{u}\in\mathbb{R}^d}{\operatorname{argmin}} \left\| \Lambda \cap (\mathcal{S} + \mathbf{u}) \right\|^2 \text{ where } \left\| \mathcal{C} \right\|^2 = \sum_{k=1}^{N} \left\| \mathbf{c}_k \right\|^2 . \tag{3.34}$$

Method I leads to truncations that are natural for the chosen lattice, truncations were the outmost shell is full. This can of course only be achieved for certain values of the number of VQ points. Method II, III and IV can give arbitrary VQ sizes. Method IV has been used by Conway and Sloane [33] in a different application, and they also propose an iterative algorithm to perform the energy minimization. The first and second method (I and II) have proved best in the cases tested in this study. Since only a limited set of rates can be achieved with method I, method II is preferred in this paper, although some results with method I are also reported.

After the truncation, the lattice VQ should be scaled to give the best possible performance. The scale factor can be approximated by use of high rate theory (see section 3.2), but to get better results an iterative procedure is often necessary, were the optimal scaling is found for a training database. Several authors have previously studied lattice scaling by iterative procedures, e.g., [8, 30, 34]. In [30], lattice VQ of iid Gaussian and Laplacian is treated, and the scaling is done by numerical optimization.

Data-optimized truncation: In applications, the source pdf is generally not analytically known, but described by an empirically collected database. In this case, we propose a data-optimized truncation, where every vector in the database is classified to its closest point in the full lattice, and the most probable lattice points are kept in the lattice quantizer. In contrast to truncation for known distributions, there is no way to avoid storing the truncation information for the data-optimized truncation. The algorithm is described in the following steps:

Step 1: An approximate scaling of the chosen lattice must be found. For iid Gaussian pdfs, and for pdfs that can be approximated as iid Gaussian, the high-rate scaling formulas in section 3.2 can be used. For unknown pdfs, ad-hoc scaling may be necessary. We have used a scaling rule that makes the granular distortion of the lattice equal to the distortion of a pdf-optimized quantizer with the desired rate, according to the Gaussian lower bound D_{GLB} (2.13) in section 2.3:

$$s = \sqrt{\frac{D_{\text{GLB}}}{G}}, \tag{3.35}$$

where G is the quantization constant of the lattice. The estimated scale factor is only an approximation of the optimum scale, but the truncation procedure is not very sensitive to the scale, and mismatches are easily detected in step 3 of this algorithm. In all tested cases, this method has proven sufficient.

Step 2: Classify each vector in the database to the nearest lattice vector, by use of a nearest-neighbor algorithm for the chosen lattice [9]. The lattice points with the N highest probabilities become codevectors in the lattice quantizer.

Step 3: An optimal scale factor s^* for the lattice quantizer is found, by some numerical optimization method. If the scale factor is very different from the one found in step 1, go to step 2 and repeat the procedure using the new scale factor s^*.

Index-optimized truncation: In [33], Conway and Sloane introduce *Voronoi codes*, where the truncation is chosen as an integer multiple of the Voronoi region of the lattice. Forney subsequently generalizes the concept to other truncation shapes in [35]. With the Voronoi codes, the indexing of the lattice VQ is greatly simplified. However, the Voronoi code truncation is generally not optimized for the pdf, and performance loss may result[5].

3.5 Indexing

In addition to the choice of Λ and $\mathcal{S}$, lattice VQ design involves one more issue; assignment of indices to the codevectors. This enumeration can be made aiming at several, partly conflicting, goals: *(i)* Memory saving. The indexing should have a mathematical formulation that is more compact than a full table. *(ii)* Fast encoding. The indexing should, in combination with one of the search algorithms that have been developed for lattices [9], yield a fast encoder $\mathcal{E}$. *(iii)* Fast decoding. The codevector should be rapidly retrievable from the index in the decoder $\mathcal{D}$. *(iv)* Symmetry. Characteristic for a lattice is that all points are alike in relation to the surrounding points. The indexing should preserve this property. In chapter 5, where an adjacency table is needed, the symmetry solves the memory problem. *(v)* Robustness. If the codebook is used for a noisy channel, bit errors should cause as little distortion as possible.

There exists an elegant solution of the indexing problem for Voronoi codes [33] in such a way that differences in indices reflect the relative position between codevectors. The method, based on modular arithmetics, satisfies *(i)*–*(iv)* above. On the other hand, Voronoi codes can only attain certain rates R, namely, those for which 2^R is an integer.

For a Gaussian probability density function, or other densities with rotational symmetry, it is beneficial if the truncation shape is as spherical as possible.

[5] Eyuboğlu and Forney show in [29] that the performance loss is small for large dimensions.

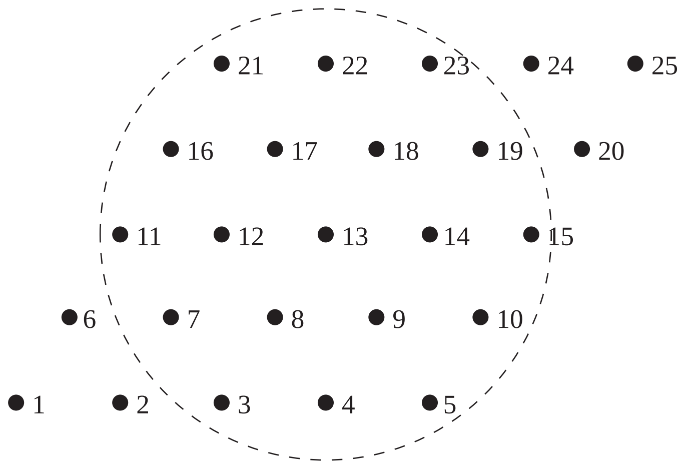


Figure 3.5. A 19-point lattice VQ, enumerated by using a 25-point set.

Unfortunately, the d-sphere does not, in general, possess any of the appealing properties mentioned above. To combine a shape that is suitable for the source (such as the d-sphere for Gaussian data) with one that has a nice indexing (such as a Voronoi region), the former can be inscribed into the latter. This approach amounts to designing a larger set that includes the codebook, enumerating this larger set, and then disregarding the points that do not belong to the codebook. For this method, *(ii)–(iv)* above are satisfied. The larger set can for instance be chosen as a Voronoi code [33]. An alternative larger set is $\mathbf{B}^{\mathrm{T}} \cdot \mathbf{z}$, where $\mathbf{z}$ is a rectangular subset of the d-dimensional cubic lattice. Figure 3.5 illustrates the latter method for a 2-dimensional example, where a 19-point lattice VQ is enumerated by using a 25-point set, for which *(ii)–(iv)* are satisfied. In the VQ design algorithm in chapter 4 and 5, we employ an indexing method in this category.

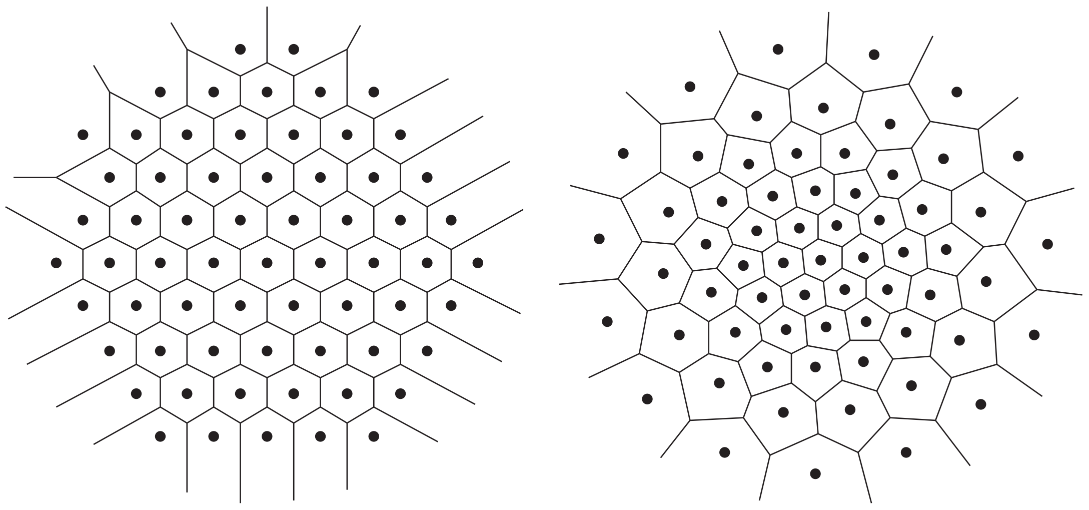


Figure 3.6. Two 64-point quantizers for a Gaussian pdf. Left: a lattice VQ. Right: a well-trained VQ.

3.6 Lattice VQ examples

In figure 3.6, a lattice VQ and a pdf-optimized VQ are depicted. The SNR values for the lattice quantizer and the optimized quantizer are 14.6 dB and 15.3 dB, respectively. In figure 3.7, the performance of lattice VQ is compared to pdf-optimized VQ for a 2- and a 5-dimensional iid Gaussian pdf. As predicted by the lattice high rate theory, the discrepancy between lattice VQ and pdf-optimized VQ increases for higher rates. More results for lattice quantization of Gaussian variables in 2 to 5 dimensions are reported on in section 6.2.

If the pdf-trained VQ in figure 3.6 is studied in detail, a feature of high rate quantizers can be observed: the structure is well-ordered, and the environment of the VQ points is locally similar to a lattice, at least for the points close to the center. This feature is exploited in the next chapter, to design VQs for fast search.

4. LATTICE-ATTRACTED VQ DESIGN

In this chapter, we propose an extension to standard VQ design algorithms, a *lattice-attracted* design algorithm, where the codebook is initialized with a truncated lattice, and the codevectors are updated to maintain a local lattice similarity for each iteration. The goal with this procedure is to make it possible to exploit the local lattice-similarity for fast nearest-neighbor search.

A sketch of a lattice-attracted algorithm is described in the following steps:

I: Initialize the VQ with a truncated lattice. An adjacency table for the lattice is also required, denoted the *lattice adjacency table*. This table consists of all neighbors to codevector **0** (vector zero), together with rules to compute the neighbors to an arbitrary point in the lattice.

II: Train the VQ with a conventional design method, but add procedures to approxi-

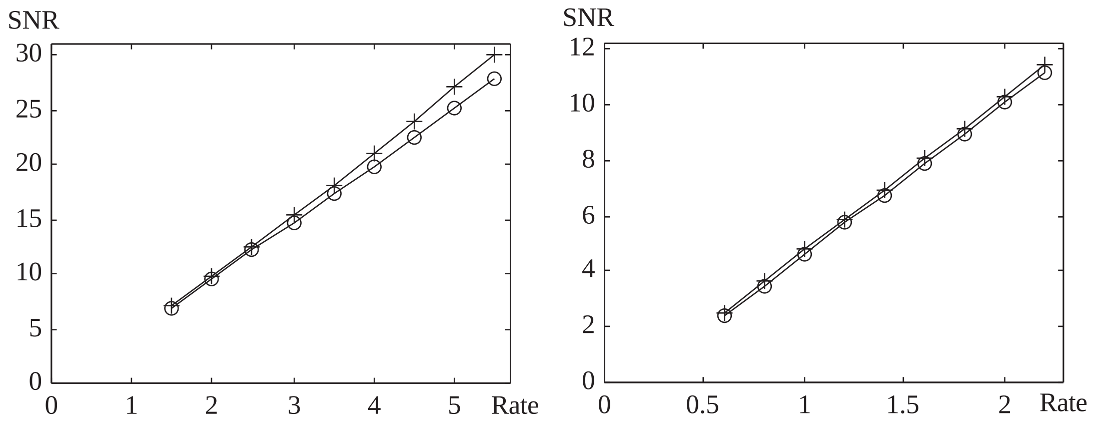


Figure 3.7. SNR as a function of rate for lattice VQ (○) and pdf-optimized VQ (+). Left: 2-dimensional VQ. Right: 5-dimensional VQ.

mately keep the initial set of neighbors, as defined by the lattice adjacency table.

The initialization procedure is described in section 4.1. In sections 4.2 and 4.3, we study how to extend two standard design algorithms, the generalized Lloyd algorithm [36] and a competitive learning algorithm [37], to approximately keep a predefined neighbor structure. In chapter 5, a novel lattice-based nearest-neighbor search method is described, based on the local lattice-similarity of the VQs trained with the proposed lattice-attracted algorithm. It is even possible to apply the fast search method during the training, as described in section 5.2.

The algorithm introduced here can, together with the specialized fast nearest-neighbor search method described in chapter 5, be viewed as a link between lattice quantization and unconstrained quantization, with the goal to combine some of the advantages of both methods.

4.1 Lattice initialization

Most iterative VQ design algorithms, such as the generalized Lloyd algorithm [36][6], or the competitive learning algorithm [37], can easily be trapped in a local distortion minimum when seeking the global minimum. A well-chosen initialization procedure can help the algorithm to avoid local minima far from the global minimum. For example, the generalized Lloyd algorithm is often initialized by a splitting procedure, proposed by Linde et al [3] (the LBG algorithm). Another possibility is to initialize the VQ with a truncated lattice. Here, we use the lattice as a good initialization for further training, but also to find a lattice adjacency table for use in the fast search procedures described later.

The lattice initialization procedure starts with selection of a lattice with a good quantization constant G, as discussed in section 3.3. The lattice is truncated by any of the methods described in section 3.4. If the pdf of the source process is given by a database, the data-optimized truncation procedure can be used. For known pdfs, the lattice can be truncated by an iso-probability contour.

Now an adjacency table must be found for the chosen lattice. Voronoi neighbors of some standard lattices can be found in [9]. As discussed in section 3.1, the neighbors to a codevector can be computed by translation of the neighbors to any other codevector, so only neighbors to the zero codevector have to be stored. A simple enumeration technique is discussed in section 3.5, where the lattice VQ is enumerated by using a larger set with desirable properties. A possible larger set is given by $\mathbf{B}^{\mathrm{T}} \cdot \mathbf{z}$, where $\mathbf{z}$ is a rectangular subset of the cubic lattice. The technique is illustrated in figure 3.5, where we see that the neighbors to an arbitrary point in the lattice VQ can be found by

[6] The generalized Lloyd algorithm is a direct generalization of a work by Lloyd, first presented in an unpublished technical note, “Least squares quantization in PCM”, at Bell Labs 1957.

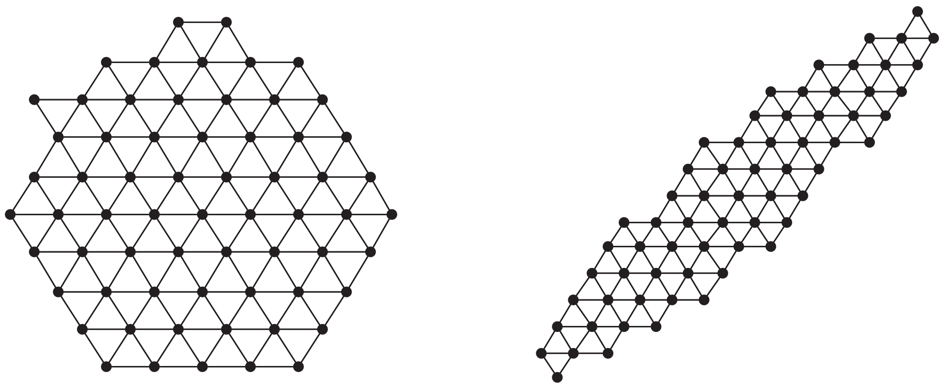


Figure 4.1. Neighbor structure (lines) for two lattice VQs (dots). Left: A lattice VQ optimized for uncorrelated Gaussian data. Right: A lattice VQ optimized for correlated Gaussian data, $\rho = 0.9$.

adding an offset of ±1, ±4 or ±5 to the index of the point. This is not the most efficient method in terms of required storage, but it works and it is simple. A more storage-efficient larger set is the Voronoi codes discussed in section 3.5 and [33, 35], and these have been used in table 6.7. With the larger-set methods above, the neighbors to the actual codevector are found by a simple procedure; the index of the codevector is found, the offset to the wanted neighbor is added, and the codevector corresponding to the neighbor index is found[7]. The first operation, finding the index of a codevector, can be solved by storing a table of indices, with one integer index for each codevector. Adding offset is trivial, and finding the codevector corresponding to the neighbor index is either solved by looking in the index table, or in another table with index-to-codevector translations (or by a compromise between those two alternatives). See section 6.4 for storage requirements of the translation tables, and overhead complexity of the translation.

An alternative to ellipsoid truncation and larger-set indexing by table look-up, is direct use of the Voronoi codes in [33], for which no translation tables are necessary. However, a Voronoi-shaped truncation region is in general not optimal for the source pdf, and performance loss results.

For a complete description of the lattice Voronoi region, the distances to the neighbors are also stored. The set of neighbors to the zero codevector, together with the corresponding distances, describes the Voronoi region of any point in the lattice.

The features of the lattice initialization procedure are here illustrated by examples of two-dimensional vector quantizers. In figure 4.1, two 64-point VQs are plotted, directly after being initialized with a truncated lattice. Each VQ point and its neighbors, according to the lattice adjacency table, are connected by lines. The regular structure of the lattice initialization is clearly visible.

[7] Some of the codewords will not have a full set of neighbors, due to the truncation of the lattice. Missing neighbors are easily detected with the table look-up methods used here.

In the following sections, we will try to optimize the quantizers for the given source, while still maintaining a locally lattice-similar structure. The neighbors according to the lattice adjacency table, denoted the *lattice neighbors*, will deviate from the true Voronoi neighbors of the quantizer, but large similarities will remain, if the optimization procedure is successful.

4.2 Lattice attraction for the generalized Lloyd algorithm

The generalized Lloyd algorithm (GLA) is often used for unconstrained VQ design. In GLA, the two necessary conditions, (2.6) and (2.7), are alternatingly iterated until the quantizer has converged. GLA is a greedy algorithm, with the feature that the average distortion decreases for each iteration. This means that GLA finds the nearest local minima, and stops the iteration. To overcome this behavior, many methods have been proposed on how to add randomness to GLA [38], in order to make it possible to evade local minima. A good initialization is of prime importance for the success of GLA.

GLA is briefly described in table 4.1, step 1–3 and 5. To extend GLA to maintain the neighborhood structure as given by the lattice adjacency table, we add an extra step (step 4 in table 4.1), where all codevectors are moved a small step to increase the local lattice-similarity. This extra step can be implemented in several ways, and we describe one such way below. In advance, the codebook is initialized with a truncated lattice, and a lattice adjacency table is found, as described in section 4.1. After the standard GLA iteration, each codevector is moved a short step towards the centroid of its neighbors, according to the distance to the corresponding neighbors in the lattice we want to mimic. In this way, the geometrical environment to each point in the VQ becomes more similar to the lattice, but each point has still a high degree of freedom during the training. The algorithm, from now on denoted *lattice-attracted GLA* or LA-GLA, is described in table 4.1, where step 4 is added to a standard GLA. In this algorithm description, the function to compute the lattice neighbors is denoted $\mathcal{N}(k,i)$, giving neighbor k of codeword i in the codebook. With l_k, we refer to the distance to neighbor k in the chosen lattice.

The step size parameter ε_m can be chosen to be constant over the training phase, or it can be a function of time. We have experimented with a linearly decreasing (to zero) step size,

$$\varepsilon_m = \varepsilon_0 \cdot \left(1 - \frac{m}{M}\right), \tag{4.1}$$

where ε_0 is the start step size and M is the total number of iterations of the algorithm. This choice makes the lattice attraction weaker and weaker, and at the end there is no attraction at all. We have experimented with different initial step sizes, and found that a

Table 4.1. The lattice-attracted GLA algorithm.

Step 1. Initialize the codebook $\mathcal{C}_1 = \left\{\mathbf{c}_1^{(1)}, \mathbf{c}_2^{(1)}, \ldots, \mathbf{c}_N^{(1)}\right\}$. Set $m = 1$.

Step 2. For the given codebook $\mathcal{C}_m$, classify each vector $\mathbf{x}$ in the training database $\mathcal{T}$ to a region $\Psi_k^{(m)}$, using the nearest neighbor partition

$$\Psi_k^{(m)} = \left\{\mathbf{x} \in \mathcal{T} : \left\|\mathbf{x} - \mathbf{c}_k^{(m)}\right\|^2 \le \left\|\mathbf{x} - \mathbf{c}_i^{(m)}\right\|^2 \text{ for all } i \in (1, N)\right\}$$

If a tie occurs, that is, if $\left\|\mathbf{x} - \mathbf{c}_k^{(m)}\right\|^2 = \left\|\mathbf{x} - \mathbf{c}_i^{(m)}\right\|^2$ for one or more i, assign $\mathbf{x}$ to the region $\Psi_i^{(m)}$ for which i is smallest.

Step 3. Compute a new codebook using the centroid condition

$$\mathbf{c}_k^{(m)} := \frac{1}{\left|\Psi_k^{(m)}\right|} \sum_{i=1}^{\left|\Psi_k^{(m)}\right|} \mathbf{x}_i$$

where the sum is over all training vectors $\mathbf{x}$ classified to $\Psi_k^{(m)}$, and $\left|\Psi_k^{(m)}\right|$ is the cardinality of the set $\Psi_k^{(m)}$ (the number of elements in $\Psi_k^{(m)}$). If $\left|\Psi_k^{(m)}\right| = 0$ for some k, use some other code vector assignment for that cell.

Step 4. Move all codevectors a small step ε_m to increase the lattice similarity,

$$\mathbf{c}_i^{(m+1)} = \mathbf{c}_i^{(m)} + \varepsilon_m \cdot \sum_{k=1}^{K(i)} \left(1 - \frac{w(\mathcal{N}(k,i))}{\left\|\mathbf{c}_{\mathcal{N}(k,i)}^{(m)} - \mathbf{c}_i^{(m)}\right\| / l_k}\right) \cdot \left(\mathbf{c}_{\mathcal{N}(k,i)}^{(m)} - \mathbf{c}_i^{(m)}\right) \qquad i = 1, \ldots, N,$$

where $\mathcal{N}(k,i)$ is the lattice adjacency function, $K(i)$ is the number of neighbors to codeword i, and $w(j)$ is the average weighted distance between a codevector k and its neighbors,

$$w(j) = \frac{1}{K(j)} \cdot \sum_{k=1}^{K(j)} \left\|\mathbf{c}_{\mathcal{N}(k,j)}^{(m)} - \mathbf{c}_j^{(m)}\right\| / l_k .$$

The new set of vectors defines a new codebook, $\mathcal{C}_{m+1} = \left\{\mathbf{c}_1^{(m+1)}, \mathbf{c}_2^{(m+1)}, \ldots, \mathbf{c}_N^{(m+1)}\right\}$.

Step 5. Stop the iteration if some stopping criterion has been reached, for example if the average distortion for $\mathcal{C}_{m+1}$ has changed by a small enough amount compared to the distortion of $\mathcal{C}_m$. Otherwise, set $m := m + 1$ and go to step 2.

value of ε_0 in the interval $0.05 - 0.1$ leads to good performance. The extra step is performed only once per iteration of the full training database, and thus the extra complexity is small.

In figure 4.2, two 64-point quantizers are depicted after being trained for a jointly Gaussian distribution with the LA-GLA algorithm, where the codebooks were initialized as in figure 4.1. We see that most of the lattice neighbor structure is retained, but that the quantizers are more optimized for the Gaussian pdf now. Results from simulations with the LA-GLA method are reported on in section 6.3.

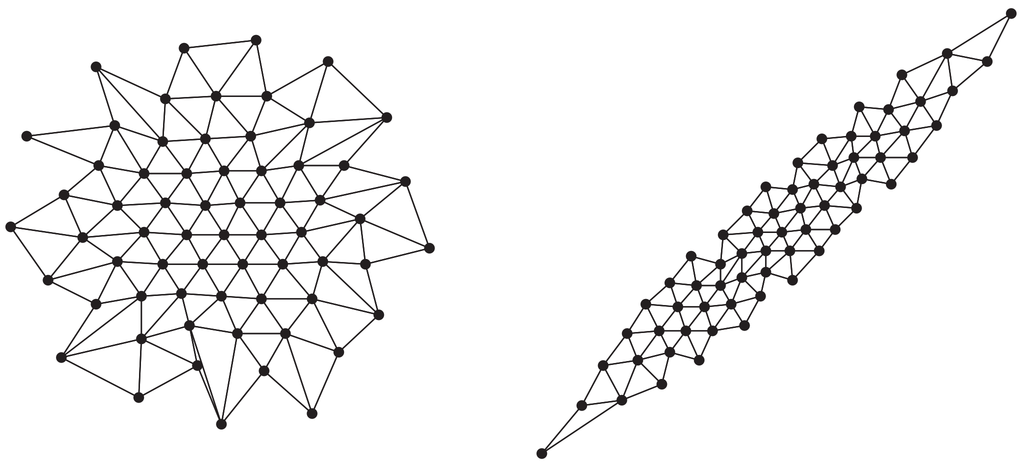


Figure 4.2. Two VQs optimized for Gaussian data, trained with the LA-GLA algorithm. Lattice neighbors are depicted as lines, and codevectors as dots. Left: uncorrelated data. Right: correlated Gaussian data, $\rho = 0.9$.

4.3 Competitive learning with lattice attraction

Competitive learning (CL) [37] was first developed for training of artificial neural networks, but can also be used for vector quantization training. In the CL algorithms, the training vectors are presented one by one, and only one codevector (the closest one) is adjusted for each input vector. The learning rule of CL can be derived from the two necessary conditions in section 2.2 [39], which make CL and GLA essentially equivalent. The main difference is that GLA works in a batch mode, were all training vectors are presented before the codevectors are adapted, as opposed to the sample iterative technique used in CL algorithms. Another important difference is that in contrast to GLA, the CL algorithm is not greedy; the average distortion does not necessarily decrease at each iteration. This allows the CL algorithm to evade some local minima.

In [37], Kohonen presents the *self-organizing feature map*, which extends CL by modifying not only the winner at each iteration, but also neighbors to the winner according to some topological map. The map is often a two-dimensional square lattice, where the neighbors can be easily computed. A feature of Kohonen training is that the structure of the map is imposed on the quantizer. Knagenhjelm [40] uses a *Hamming map*, in order to train VQs where the Hamming distance between codewords and the Euclidean distance between codevectors are closely related. This is shown to substantially robustify the VQ for transmission over a noisy binary symmetric channel.

The self-organizing feature map is a straightforward way to attract the quantizer to the lattice. The neighbors in the map are given by the lattice adjacency table, and the winning candidate is modified together with all neighbors in the table for each presentation of input data. The algorithm is described in table 4.2. The neighbor step size ε_m is, as in the LA-GLA, linearly decreasing,

Table 4.2. The competitive learning algorithm with a lattice topology map.

Step 1. Initialize the codebook $\mathcal{C}_1 = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}$. Set $m = 1$.

Step 2. A random vector $\mathbf{x}_m$ is drawn from the training database. For the input data $\mathbf{x}_m$, find the winning candidate according to the quadratic error criterion,

$$\mathbf{c}^* = \underset{\mathbf{c} \in \mathcal{C}_m}{\operatorname{argmin}} \|\mathbf{x}_m - \mathbf{c}\|^2.$$

Step 3. Modify the winning codevector as

$$\mathbf{c}^* := \mathbf{c}^* + \eta_m \cdot (\mathbf{x}_m - \mathbf{c}^*).$$

where the "temperature" η_m is linearly decreasing from an initial temperature η_0:

$$\eta_m = \eta_0 \left(1 - \frac{m}{M}\right).$$

Step 4. Modify the neighbors to the winning candidate a small step ε_m, according to

$$\mathbf{c}_k := \mathbf{c}_k + \eta_m \cdot \varepsilon_m \cdot (\mathbf{x}_m - \mathbf{c}_k) \;\; , k = 1, ..., K.$$

where $\mathbf{c}_k$ is one of the totally K neighbors (found in the lattice adjacency table) to $\mathbf{c}^*$.

Step 5. If $m = M$, then stop the iteration. Otherwise, set $m := m + 1$ and go to step 2.

$$\varepsilon_m = \varepsilon_0 \left(1 - \frac{m}{M}\right). \tag{4.2}$$

The resulting CL algorithm is denoted the *lattice-attracted competitive learning* (LA-CL) algorithm. Results of simulations with this algorithm are presented in chapter 6.

5. FAST SEARCH OF LATTICE-ATTRACTED VQ

In [11], an algorithm for fast search of arbitrary VQs is described. With this algorithm, denoted the *steepest neighbor descent* (SND) algorithm, an adjacency table is precomputed, consisting of all Voronoi neighbors to all codevectors in the VQ (how to find the adjacency table is described in [11]). When the table is found and stored, the actual quantization can begin. For each input vector $\mathbf{x}$, one of the codevectors in the codebook is selected as a starting hypothesis $\mathbf{c}^{(0)}$. The distance between $\mathbf{x}$ and $\mathbf{c}^{(0)}$ is computed, and then the distances between $\mathbf{x}$ and the neighbors to $\mathbf{c}^{(0)}$ (found in the adjacency table) are computed. When all neighbor distances have been computed, the neighbor closest to $\mathbf{x}$ becomes the new hypothesis $\mathbf{c}^{(1)}$.

This procedure is repeated until a hypothesis vector is found whose neighbors are all worse. It can easily be shown that when a codevector with lower distance to the input vector than all its neighbors is found, this vector is the optimal codevector (see (2.10)).

The main disadvantage of the SND algorithm is the storage requirements for the precomputed adjacency table, typically many times the required storage of the codebook. For example, a 12 bit 6-dimensional VQ requires around 700 kbyte storage for the adjacency table [11], and this is impractical for many applications.

Lattices have a feature that can be exploited to reduce the storage requirements for the SND algorithm; all neighbors to an arbitrary point in a lattice can be found by translation of the neighbors to the zero lattice point. To find the neighbors to an arbitrary point in a lattice VQ, the neighbors to the zero point are translated, and the set of neighbors is truncated by the global truncation rules. Thus, we can apply the SND algorithm to a lattice VQ, supported only by the neighbors to a single region. However, this would not be a very competitive algorithm, since fast specialized search algorithms have been developed for many important lattices [33]. A better choice is to apply the low-storage SND algorithm to the well-performing lattice-attracted quantizers from chapter 4. These quantizers are trained to maintain a lattice neighbor structure, and are well suited for low-storage SND search.

In this chapter, we discuss how to apply the steepest neighbor descent method to the quantizers trained by LA-GLA or LA-CL algorithm.

5.1 An extended SND algorithm

Here, we will propose an SND algorithm to suit the lattice-attracted quantizers from chapter 4. The lattice neighbors of the lattice-attracted quantizers (c.f. figures 4.1 and 4.2) are not always in perfect correspondence with the real Voronoi neighbors. False neighbors, i.e., codevectors listed as lattice neighbors without being Voronoi neighbors, constitute no problem, but not listed Voronoi neighbors can lead to erroneous decisions, and must be considered.

An important issue is the starting point of the algorithm, i.e., the choice of an initial hypothesis codevector. For the tested Gaussian densities, the trained lattice-attracted quantizers show a high degree of similarity with the lattice quantizer used for the initialization of the LA-GLA and LA-CL algorithms; the codevectors stay in general fairly close to their initial positions. Thus, a good starting hypothesis is the vector found by nearest-neighbor search of the initial lattice quantizer. For many important lattices, nearest neighbor search can be done with very low complexity [9]. No extra storage is required for this, just a search algorithm for the chosen lattice.

We have extended the SND algorithm to handle the special problems with an incomplete adjacency table, and also to exploit the lattice-similarity to find a good starting point. Three extensions have been used:

I An initial hypothesis is found by nearest-neighbor search of the chosen lattice.

II If the current hypothesis codevector is closer to the input vector than all of its neighbors, the neighbor descent search continues from the second best vector. This procedure is repeated until no improvement is obtained.

III When the SND terminates and declares a winning codeword, an *exception table* is consulted, including Voronoi neighbors not found in the lattice adjacency table. If the winning codeword is found in the exception table, the listed extra neighbor(s) is also tested.

The exception table should be constructed prior to the actual quantization. All the missing Voronoi neighbors do not have to be included in the exception table, only those that lead to a substantially higher distortion if not included. The exception table can be found by running a full search in parallel with the SND search for a training database, and observing when the answers from the two search procedures differ.

The first extension requires a lattice nearest-neighbor search prior to the VQ search. The complexity of this extension varies with the effectiveness of the search algorithms for the actual lattice, but for the lattices used here, the complexity corresponds to 0.5–2 extra distance computations. No extra storage is needed. The second extension has experimentally shown to lead to a few additional distance computations for each input vector, compared to the standard SND algorithm, but no extra storage is required. The third extension, the exception table, requires some extra storage, but the extra search complexity is small, since the exception table is seldom consulted.

Experiments show that if the performance loss compared to a full search is required to be less than 0.01 dB, the exception table can be very small, typically a few entries for the 2-dimensional VQs tested here, and 20–30 entries for the high rate 5-dimensional VQs. If no performance loss at all is allowed, the 5-dimensional VQs may require an exception table that includes up to 10–15% of the vectors in the codebook, to compensate for all missing neighbors, even though these occur with a probability close to zero.

If the exception tables are excluded, some performance loss is inevitable. The 5-dimensional VQs require larger exception tables to reach 0.01 dB performance loss than the 2-dimensional VQs, but on the other hand, if the exception tables are excluded, the performance loss of the 5-dimensional VQs is small, for the tested VQs always less than 0.05 dB. In section 6.4, we report the performance, in terms of storage and search complexity, for quantizers where the exception table is designed for "almost lossless" (less than 0.01 dB loss) operation.

Table 5.1. The extended steepest neighbor descent (eSND) algorithm.

Step 1: Find an initial hypothesis codevector $\mathbf{c}^*$, by a lattice nearest-neighbor search. Set the temporary codevector $\mathbf{c}$ to null.

Step 2: Find the lattice neighbors to $\mathbf{c}^*$, by look-up and translation of the lattice adjacency table.

Step 3: Compute the distortion of all untested neighbors. If a better codevector than $\mathbf{c}^*$ is found, this becomes the new hypothesis $\mathbf{c}^*$, and the execution continues at step 2. If no better neighbor can be found, continue to step 4.

Step 4: If the current hypothesis $\mathbf{c}^*$ is equal to the temporary codevector $\mathbf{c}$, continue to step 5. Otherwise, set the temporary codevector $\mathbf{c}$ to the second best codevector found up to then, set $\mathbf{c}^* = \mathbf{c}$, and go back to step 2.

Step 5: If the current best hypothesis is listed in the exception table, compute the distortion of the extra neighbor(s) as given by the exception table.

Step 6: The best codevector found until now is returned.

The extended SND algorithm (eSND) is described in table 5.1.

The algorithm works well for Gaussian data. An interesting question is how well it generalizes to other pdfs. The simple answer is that it generalizes to pdfs that can be well quantized using a quantizer with locally lattice-similar structure. These include pdfs where direct lattice quantization works well, and thus the VQ points typically move only a small distance from the lattice initialization. It also generalizes to pdfs for which a multidimensional compander in combination with a lattice quantizer works well (see, e.g., [41] for a treatment of this subject). However, the question if the algorithm works well for arbitrary pdfs is a subject for further research.

In section 6.4 we report on the search complexity reduction that can be achieved with the eSND algorithm. In section 5.2, we study how to apply the eSND algorithm already during the design phase, with a design complexity reduction as result.

5.2 Fast search during the design phase

To speed up the design procedure by the LA-GLA and LA-CL algorithms, the fast search procedure can be incorporated in the training. The introduction of the eSND search during the design phase leads to a few problems. First, the exception table in the eSND algorithm must be constructed ”on-line” during the design process. The exception table during design may be far from complete; the training has experimentally shown to be fairly insensitive to a few misclassifications. We have experimented with construction of an exception table after the first iteration of the GLA algorithm, by doing a full search in parallel with the eSND. For the following iterations only eSND search is performed. After some iterations, it might be necessary to reconstruct the exception table.

Another problem we encountered in the development of the LA-CL method was a break-down tendency (failure to improve the VQ) for high initial temperatures η_0. This is caused by the random reordering of codevectors that occur for high temperatures, destroying the well-ordered initial lattice structure. When the lattice structure is destroyed, the eSND search fails more often to find the optimal codevector, and as a result the VQ is adapted to destroy the lattice structure even more. However, the break-down temperature is distinct and well above realistic start temperatures, so the problem is easily avoided. The LA-GLA algorithm has not shown any tendencies to break down for the problems treated in this report.

5.3 Related work

In the literature, some other reports on fast search for unconstrained VQs can be found. As discussed earlier, there are some methods based on the neighbor descent concept. These algorithms show similar performance as the proposed eSND algorithm for lattice-attracted VQs, but the storage requirement for the adjacency table is typically many times the required storage of the codebook [10, 11]. In [42], only a fraction of the full adjacency table is stored, with a suboptimal search procedure as a result.

Another method is the *K-d tree* technique, proposed in [43], and further developed in, e.g., [13]. A binary tree, with hyperplane decision tests at each node, is precomputed and stored. The decision tree leads to one of a set of terminal nodes, where small sets of still eligible candidate vectors are listed.

In the *projection* technique [44], a rectangular partition of the space is precomputed and stored. During the search, the rectangular cell containing the input vector is found, and the distances to a small number of eligible codevectors are computed. The number of distance calculations with this method is typically very small, but the overhead complexity is considerable.

Anchor point algorithms [12, 45] are algorithms where VQ points are excluded from the search by use of the triangle inequality. The distances from a small set of anchor points to each of the codevectors are precomputed and stored. The encoder then computes the distance between the input vector and each anchor point, and a large number of codevectors can be eliminated from the nearest neighbor search.

In [46], a Kohonen feature map is used as a basis for a fast search algorithm. However, the search algorithm shows poor performance, with a high percentage of misclassifications, due to the selection of a map that is not a good quantizer in itself.

For comparison, we have included measurements of an anchor point algorithm and the projection technique, in section 6.4.

6. EXPERIMENTS

In many real-world applications employing vector quantization, the Gaussian distribution is used as a model for the incoming data, and also as a model of the quantization error. This is mainly because it is possible to theoretically compute important parameters for Gaussian pdfs, but also because the Gaussian distribution is often a good approximation to the pdf of the actual data. This makes the performance of quantization of Gaussian variables interesting.

In this chapter, we present simulation results of lattice quantization and lattice-attracted VQs, and study their performance for Gaussian pdfs. In section 6.1, we describe the databases used in the experiments. In section 6.2, the performance for lattice VQ of Gaussian data is given, and in section 6.3, the performance of the new lattice-attracted method is tabulated. The achievable search complexity reductions and extra memory requirements for the eSND method are given in section 6.4, where it is also compared to an anchor point algorithm.

6.1 Databases

All Gaussian variables are generated by the Box-Müller method, using a well-tested random number generator from [47]. Both correlated and uncorrelated databases are generated. The correlated data are sequences of samples, drawn from a first order Markov process with correlation coefficient $\rho = 0.9$.

6.2 Results for Gaussian variables

In this section, we present the performance of lattice quantization of Gauss-Markov processes. The lattices are truncated as described in section 3.4, with method II for known pdfs, and the optimal scale factors are determined by an iterative procedure, using a database of 200 000 samples. For comparison, we also present SNR values for optimized Gaussian vector quantization (20 million iterations of a CL algorithm are used to train the quantizers). For the performance evaluation, an independent evaluation database with 1 million Gaussian vectors is used, both for lattice VQs and pdf-optimized VQs.

In table 6.1, we present signal-to-noise-ratios (SNR) for quantization of an iid Gaussian pdf[8]. We see that lattice quantization can give competitive performance for low and medium rates, but for higher rates, the pdf-optimized VQ is significantly better. As predicted by the high-rate lattice theory in section 3.2, a lattice quantizer is inferior to a pdf-optimized quantizer when the rate is high.

[8] Note that the results for high-rate pdf-optimized quantizers show signs of undertraining; especially the SNR values for 2 dimensions, 2048 codewords could be improved with longer training.

Table 6.1. SNR (in dB) for lattice VQ and pdf-optimized VQ (inside parenthesis), for quantization of uncorrelated Gaussian vectors.

Number of codewords	Dimension of VQ			
	***d*=2**	***d*=3**	***d*=4**	***d*=5**
8	6.78 (6.96)	4.29 (4.48)	3.16 (3.34)	2.38 (2.53)
16	9.48 (9.68)	6.20 (6.29)	4.41 (4.67)	3.48 (3.66)
32	12.09 (12.44)	7.91 (8.10)	5.90 (5.99)	4.59 (4.77)
64	14.64 (15.29)	9.68 (9.95)	7.17 (7.36)	5.76 (5.84)
128	17.22 (18.18)	11.48 (11.83)	8.54 (8.75)	6.77 (6.93)
256	19.85 (21.10)	13.24 (13.74)	9.90 (10.15)	7.89 (8.05)
512	22.47 (24.04)	14.97 (15.66)	11.22 (11.57)	8.98 (9.17)
1024	25.11 (27.03)	16.71 (17.62)	12.59 (13.00)	10.07 (10.31)
2048	27.75 (29.88)	18.45 (19.62)	13.91 (14.49)	11.12 (11.47)

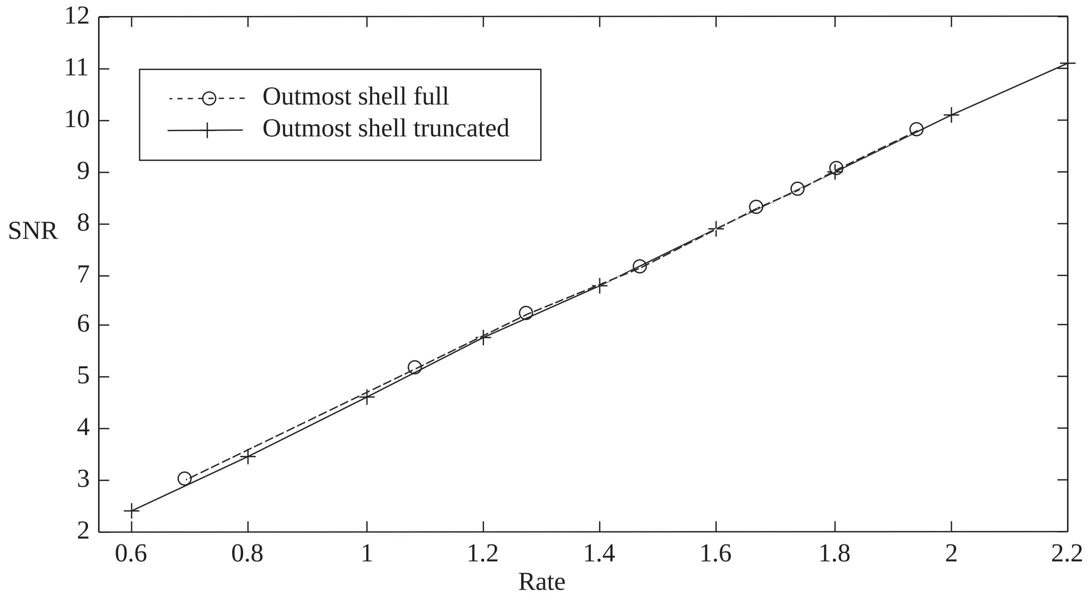


Figure 6.1. Performance for a truncated lattice VQ on a 5-dimensional iid Gaussian pdf. The crosses (x) indicate performance for lattice VQ where the number of points is truncated to an even power of two, and the circles (o) indicate the performance with a fully populated outmost shell.

We also wanted to examine the importance of the truncation procedure. For this purpose, we have applied truncations that are natural for the chosen lattice, i.e., truncations that acknowledge the shell structure of the lattice, and keep the outmost shell fully populated (method I in section 3.4). This can of course only be achieved for certain number of points. For the D_5^* lattice, the number of points in the shells[9] is, from inside out, given by the theta series {1, 10, 32, 40, 80, 160, 90, 112, 320, ...}, and thus the number of points in a quantizer with fully populated shells are {1, 11, 43, 83, 163, 323, 413, 525, 845, ...}. In figure 6.1, we compare the performance of lattice VQs with fully populated shells with VQs where the number of points is an integer

[9] Other theta series are possible if the lattice is translated.

Table 6.2. SNR (in dB) for lattice VQ and pdf-optimized VQ (inside parenthesis), for a first order Gauss-Markov process, with correlation coefficient 0.9.

Number of codevectors	**Dimension of VQ**			
	***d*=2**	***d*=3**	***d*=4**	***d*=5**
8	9.72 (10.83)	9.20 (9.37)	8.19 (8.48)	7.43 (8.09)
16	12.48 (13.55)	10.45 (11.41)	9.23 (10.20)	8.37 (9.39)
32	15.13 (16.25)	12.30 (13.21)	10.50 (11.66)	9.48 (10.69)
64	17.98 (19.05)	14.08 (15.01)	12.08 (13.03)	11.02 (11.85)
128	20.82 (21.87)	16.16 (16.85)	13.44 (14.40)	11.83 (12.96)
256	23.28 (24.81)	17.80 (18.71)	14.95 (15.77)	13.19 (14.05)
512	25.80 (27.72)	19.69 (20.60)	16.46 (17.16)	14.37 (15.14)
1024	28.63 (30.67)	21.36 (22.51)	17.69 (18.56)	15.54 (16.23)
2048	31.24 (32.82)	23.16 (24.39)	19.18 (19.97)	16.70 (17.35)

power of 2. We see that for low rates, the truncation where the outmost shell is fully populated has a performance advantage, but for higher rates the ”unstructured” truncation procedure gives equivalent performance.

In table 6.2, we present signal-to-noise-ratios for lattice quantization of a first order Gauss-Markov process with correlation coefficient 0.9. We see that for correlated Gaussian data, pdf-optimized vector quantizers have in most cases a significant performance advantage over lattice quantizers.

6.3 Lattice-attracted VQ design performance

With the new lattice-attracted VQ design methods, an interesting question is if the lattice attraction leads to loss of performance compared to unconstrained VQ training. To investigate this, the performance for quantizers trained until convergence with the different methods are compared in table 6.3. The SNR values are averaged over 20 simulations with different training databases (different seeds for the random number generator). The evaluation database consists of one million Gaussian vectors. Even though the GLA algorithm is normally aborted when the distortion change is small enough, we have here chosen to run all algorithms for a predetermined number of iterations (100 million iterations are performed in all cases, where one iteration consists of finding the closest codevector to an input vector). The chosen design time is large enough for all

Table 6.3. SNR (dB) for quantizers trained until convergence with the different methods.

dim, size, corr	**CL**	**LA-CL**	**LBG**	**LA-GLA**
d=2, N=64, ρ=0	15.30	15.30	15.27	15.27
d=2, N=64, ρ=0.9	19.05	19.05	19.03	19.02
d=3, N=128, ρ=0	11.85	11.85	11.82	11.82
d=3, N=128, ρ=0.9	16.87	16.87	16.83	16.82
d=5, N=1024, ρ=0	10.32	10.32	10.25	10.26
d=5, N=1024, ρ=0.9	16.23	16.23	16.20	16.20

the methods reach convergence, i.e., the results do not improve for longer training. The size of the training database is limited (500 000 vectors) for the batch algorithms, LBG and LA-GLA, but for the competitive learning methods, the database size is “unlimited”; a new Gaussian vector is drawn for every iteration.

Note that the CL algorithms perform slightly better than LBG or LA-GLA. A reason for the inferiority of the GLA-based algorithms is the limited training database, making the greedy GLA-based algorithms more easily trapped in local minima. From the numbers in table 6.3, we conclude that the lattice attraction does not decrease the performance of the fully trained VQ, neither for GLA nor CL. For these extremely well-trained quantizers, the lattice-constraint is mainly a question of indexing of the codevectors; for all methods, the resulting structures of the quantizers are very similar. This indicates that an indexing procedure could be applied after the training procedure to make the fast eSND search possible. However, it would then be impossible to apply the eSND during the training.

In reality, it may be impractical with the tedious train-until-convergence used above, and the database size is also often limited. A more realistic database can have a size that is only 100 times the number of codewords, and in some cases even less. In figure 6.2, we compare the different design methods for limited design time and database size. We see that the lattice-attracted design methods reach a higher SNR for a limited database size, due to the attraction to a well-ordered lattice structure, a structure that otherwise can be hard to reach for limited training times and databases. No method reaches an SNR close to the optimum 15.3 dB (table 6.3).

The results in this section seem to indicate that the CL-based algorithms should be preferred for VQ design. However, the tuning of the starting temperature for the CL

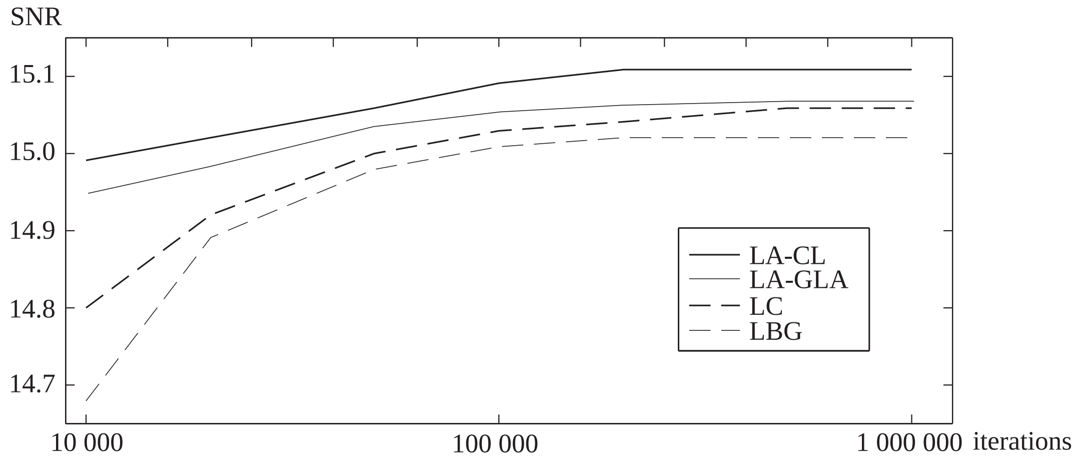


Figure 6.2. SNR as a function of number of iterations for design of a 64-point 2-dimensional VQ. For all methods, the training database contains 5000 vectors, drawn from an iid Gaussian pdf. The LBG algorithm uses a split initialization technique, while the other algorithms are initialized with a truncated lattice, giving an initial SNR of 14.6 dB.

Table 6.4. Average and maximum (within parenthesis) number of distance computations for the lattice-attracted quantizers. The database consists of uncorrelated Gaussian vectors.

Number of codewords	**Dimension of VQ**			
	***d*=2**	***d*=3**	***d*=4**	***d*=5**
8	5.8 (8)	6.1 (8)	7.7 (8)	6.7 (8)
16	7.7 (12)	10.4 (16)	12.2 (16)	12.0 (16)
32	9.1 (12)	13.8 (25)	18.7 (32)	22.1 (32)
64	9.9 (12)	17.3 (28)	24.7 (51)	32.7 (63)
128	10.6 (13)	19.6 (28)	30.9 (63)	42.1 (89)
256	10.6 (15)	21.5 (31)	36.1 (69)	53.2 (124)
512	10.5 (16)	23.3 (36)	41.5 (79)	64.8 (145)
1024	10.5 (16)	25.2 (40)	45.7 (91)	75.3 (167)
2048	10.5 (16)	25.2 (44)	50.2 (96)	81.8 (179)

algorithms can be tedious, and the empty-cluster-problem is simpler to handle in GLA-based algorithms. Thus, LBG and LA-GLA may still be preferable in some applications.

6.4 eSND performance

In this section, we report on the performance of the eSND algorithm, in terms of search complexity and storage requirements. For comparison, we have also included measurements of an anchor point algorithm, using the same databases.

Search complexity: We have applied the eSND algorithm, described in chapter 5, to quantizers trained with LA-GLA. The exception tables are designed for “almost lossless” operation, with a performance loss compared to full search that is less than 0.01 dB. The average and maximum number of distance computations are listed in table 6.4 for iid Gaussian, and in table 6.5 for Gauss-Markov ($\rho = 0.9$). The number of distance computations of a full search is of course equal to the number of codewords in the quantizer. We see that a significant reduction of the number of distance computa-

Table 6.5. Average and maximum (within parenthesis) number of distance computations for the lattice-attracted quantizers. The database consists of correlated ($\rho = 0.9$) Gaussian vectors.

Number of codewords	**Dimension of VQ**			
	***d*=2**	***d*=3**	***d*=4**	***d*=5**
8	4.3 (6)	3.8 (5)	3.8 (5)	3.9 (5)
16	6.5 (9)	6.9 (11)	7.2 (11)	7.1 (11)
32	7.9 (11)	9.8 (16)	12.4 (22)	12.7 (24)
64	9.1 (12)	13.4 (24)	15.8 (30)	16.1 (30)
128	9.9 (13)	15.7 (25)	22.5 (44)	27.1 (58)
256	10.5 (14)	19.0 (30)	28.2 (52)	33.9 (70)
512	10.7 (15)	20.8 (32)	33.1 (64)	45.8 (100)
1024	10.8 (17)	23.2 (36)	39.8 (79)	56.0 (126)
2048	10.9 (19)	24.3 (40)	44.0 (84)	66.6 (148)

tions is achieved for the eSND method, and also that the maximum is reasonable (measured for one million test vectors).

Besides of the distance computations, some additional overhead for the eSND algorithm is unavoidable. The initial hypothesis codevector is found by searching the closest vector in the lattice associated with the lattice-attracted VQ. This procedure is not very complex due to the regular structure of the lattice; for the lattices employed here, the procedure involves a rescaling of the input vector, adding an offset vector and rounding all elements towards the nearest integer. The total overhead complexity for finding the initial hypothesis is less than two extra distance computations for the lattices used here. More about lattice nearest-neighbor search algorithms can be found in [9]. There is also overhead for each distance computation. When a new hypothesis codevector is found, the lattice index of the codevector must be found, by table lookup as described in section 4.1. For each distance computation, an integer is added to the lattice index, and the codevector corresponding to the sum is found by table lookup[10]. The overhead depends on the efficiency of integer arithmetics of the given processor, but for the hardware used here (DEC Alpha), the overhead complexity is only a fraction of the complexity of the distance computations.

It is interesting to compare the eSND method with other fast nearest-neighbor search methods (see section 5.3). In comparison with other neighbor descent methods, eSND has a slight advantage, because of the good initial hypothesis given by the lattice search, but the overall performance should be similar due to the similar approaches. Among other methods, anchor point algorithms are well-known. We have implemented an anchor-point algorithm, IFAP-AESA [12]. IFAP-AESA substantially reduces the number of L2-norm distance computations, at the cost of a number of L1-norm distance computations. A procedure similar to the standard *partial distance* technique [44, 48] is employed for the L1-norm computations to further reduce the complexity. We have also implemented the projection method [44], briefly described in section 5.3. The rectangular partition is optimized for ”almost lossless” operation, with at most 0.01 dB performance loss compared to full search.

While the complexity of full search and eSND is essentially proportional to the number of L2-norm distance computations, this is not true for IFAP-AESA and the projection method. Therefore, we report the complexity in the average number of floating point multiplications, additions, comparisons and integer operations (given as a proportionality constant) per input vector. The additional overhead for eSND is described above, and for IFAP-AESA the overhead consists of frequent absolute value computations and table look-ups. The overhead complexity for the projection method

[10] If a Voronoi code is used, the table lookups are unnecessary; the indices of the codewords are given by the sorting of the codebook. But Voronoi codes may lead to performance loss.

Table 6.6. Average number of multiplications, additions, comparisons and integer operations for a full search, for an anchor point algorithm, IFAP-AESA, the projection method and for the eSND algorithm. The database consists of uncorrelated Gaussian vectors.

Dimension d,	**Multiplications, Additions, Comparisons (Integer operations)**			
VQ size N	**Full search**	**IFAP-AESA**	**Projection**	**eSND**
d=2, N=64	128, 192, 63 $(\propto N \cdot d)$	11, a=168, 117 $(\propto a)$	3, 4, 15 $(\propto N \cdot d)$	20, a=30, 20 $(\propto a)$
d=3, N=128	384, 640, 127 $(\propto N \cdot d)$	23, a=556, 386 $(\propto a)$	6, 11, 26 $(\propto N \cdot d)$	59, a=98, 39 $(\propto a)$
d=5, N=1024	5120, 9216, 1023 $(\propto N \cdot d)$	70, a=8286, 5983 $(\propto a)$	26, 47, 60 $(\propto N \cdot d)$	377, a=678, 150 $(\propto a)$

is considerably higher than for the other methods, with a large number of integer operations. Actually, the complexity of the projection method is dominated by the integer operations for the cases tested here.

The nearest-neighbor algorithms are compared in table 6.6. The number of integer operations for the projection method and for full search is proportional to the VQ size N times the dimension, while the number of integer operations for eSND and IFAP-AESA is proportional to the number of distance computations (which is the sum of L1-norm and L2-norm distance computations for IFAP-AESA). This means that the number of integer operations for IFAP-AESA and eSND grows much slower than for full search and the projection method.

We see that IFAP-AESA radically reduces the number of multiplications, but that the number of additions and comparisons remains high. IFAP-AESA can only compete with the other algorithms for hardware where the multiplication cost is dominating, but in terms of FLOPS (floating-point operations per second), IFAP-AESA is inferior. On the other hand, the projection algorithm outperforms the other algorithms in terms of FLOPS. However, as discussed above, the overhead complexity for the projection method is considerably higher, and which of the two methods that is the fastest in practice is dependent on the efficiency of the hardware.

Storage requirements: To use the eSND fast search algorithm, we must precompute and store an adjacency table, an exception table, and tables to aid translation from codebook index to lattice index and vice versa. In table 6.7, the required storage of the tables and the codebook is given for a few VQ examples. As seen in the table, the storage requirements are dominated by the codebook and the translation tables. The larger extra storage of the 5-dimensional VQ depends on that 2 instead of 1 byte is required to encode the 1024 codewords. Since we only consider unconstrained VQs, the codebook size can not be reduced, unless the precision is somehow reduced. It is possible to reduce the storage requirements for the translation tables, at the cost of extra overhead time for the eSND search.

Table 6.7. Relative and absolute storage requirements (in bytes) for examples of iid Gaussian quantization. The codebooks are stored as 4-byte floating point numbers, and the tables consist of one- or two-byte integer values. The total storage is given in percentage of codebook only storage.

Storage requirements	d=2, N=64	d=3, N=128	d=5, N=1024
Codebook	$64 \times 2 \times 4$ = 512	$128 \times 3 \times 4$ = 1536	$1024 \times 5 \times 4$ = 20480
Adjacency table	6	14	$62 \times 2 = 124$
Exception table	0	3	$15 \times 2 = 30$
Translation tables	145	371	$4149 \times 2 = 8298$
Total storage	129%	125%	141%

The anchor point algorithm requires storage of a floating point table with size $(d+1)/d$ times the size of the codebook. For the 2-, 3- and 5-dimensional cases above, the total storage, in percent of codebook only storage, are 250%, 233% and 220%, respectively.

For the projection method, a rectangular partition of the space, and a set of candidate codewords for each rectangular cell, are precomputed and stored. The total storage, in percent of codebook only storage, are 350%, 350% and 400% for the cases above.

7. SUMMARY

In this report, lattice-based quantization was studied, both from a theoretical and a practical viewpoint. Lattice-based quantization is a generalization of conventional lattice quantization, by allowing modifications of the regular lattice structure while still maintaining a local lattice-similarity.

For conventional lattice quantization, high rate theory was developed. The high rate theory leads to lattice VQ design rules, and to new insights in the performance of lattice quantization. An important conclusion was that for high rates, lattice quantization is severely inferior to optimal vector quantization. Practical solutions to problems in lattice quantization, such as truncation and scaling, were discussed, and the performance of lattice quantization of Gaussian variables was presented.

To overcome the inherent shortcomings of lattice quantization, we proposed a novel lattice-based technique for VQ design, with the feature that the resulting VQs are locally lattice-similar, but globally optimized to the input pdf. The design algorithm was complemented with a new lattice-based fast search algorithm. Experiments on Gaussian data with the proposed fast search algorithm illustrated that the performance is excellent, with only moderate extra storage requirements.

Appendix A

In this appendix, theorem I (3.13) and theorem II (3.14) in section 3.2 are proved. In section A.1, some definitions and preliminaries are presented. Section A.2 discusses the overload distortion (theorem I), and section A.3 treats the granular distortion (theorem II). In section A.4, the total distortion, which is the sum of overload and granular distortion, is treated, and methods to find the global minimum is discussed.

A.1 Preliminaries

For the proofs in the appendix, we use the definition of a *d*-sphere (3.9), the truncation radius a_{T} (3.10), and the granular region $\mathcal{G}$ (3.11), all defined in section 3.2. We also use the VQ definitions in chapter 2, and the lattice definitions in section 3.1, together with some new definitions in this section. As discussed in section 3.2, we assume zero mean, iid Gaussian variables, with unit variance samples.

A *granular Voronoi region* $\Omega_{\mathcal{G}}(\mathbf{c})$ is defined as the lattice Voronoi region Ω, translated to the codevector $\mathbf{c}$,

$$\Omega_{\mathcal{G}}(\mathbf{c}) = \Omega + \mathbf{c} = \Omega(\mathbf{c}) \cap \mathcal{G}\,, \tag{A.1}$$

where $\Omega(\mathbf{c})$ is the Voronoi region around codevector $\mathbf{c}$ (see 2.7), Ω is the lattice Voronoi region (see 3.4), and $\mathcal{G}$ is the granular region (see 3.11).

For a given input vector $\mathbf{x}$, we define $\mathbf{p}(a)$ as the closest point to $\mathbf{x}$ in a sphere with radius a,

$$\mathbf{p}(a) = \underset{\mathbf{y}:\|\mathbf{y}\|<a}{\operatorname{argmin}} \|\mathbf{x}-\mathbf{y}\| = \begin{cases} \mathbf{x} & \|\mathbf{x}\| \le a \\ a \cdot \dfrac{\mathbf{x}}{\|\mathbf{x}\|} & \|\mathbf{x}\| > a \end{cases} \tag{A.2}$$

With this definition, the distance between $\mathbf{x}$ and $\mathbf{p}(a)$ is given by

$$\|\mathbf{x} - \mathbf{p}(a)\| = \max\left(0, \|\mathbf{x}\| - a\right). \tag{A.3}$$

We define the *granular radius* $a_{\mathcal{G}}$ as the effective radius of the granular region $\mathcal{G}$,

$$a_{\mathcal{G}} = \left(\frac{\operatorname{vol}(\mathcal{G})}{\operatorname{vol}(S_d(1))} \right)^{1/d}, \tag{A.4}$$

where $S_d(\phi)$ is a sphere with radius ϕ, see (3.9). The volume of the sphere $S_d(a_{\mathcal{G}})$, called the *granular sphere*, is with this definition equal to the volume of the granular region, i.e. $\operatorname{vol}(S_d(a_{\mathcal{G}})) = \operatorname{vol}(\mathcal{G})$. The granular radius $a_{\mathcal{G}}$ and the truncation radius a_{T} are closely related, and we show in (A.24) that they are equal for infinite rates.

We define a *border region* $\mathcal{B}$ in the form of a spherical shell (see figure A.1),

$$\mathcal{B} = \left\{ \mathbf{x} \in \mathbb{R}^d : a_{\min} < \|\mathbf{x}\| \le a_{\max} \right\} \tag{A.5}$$

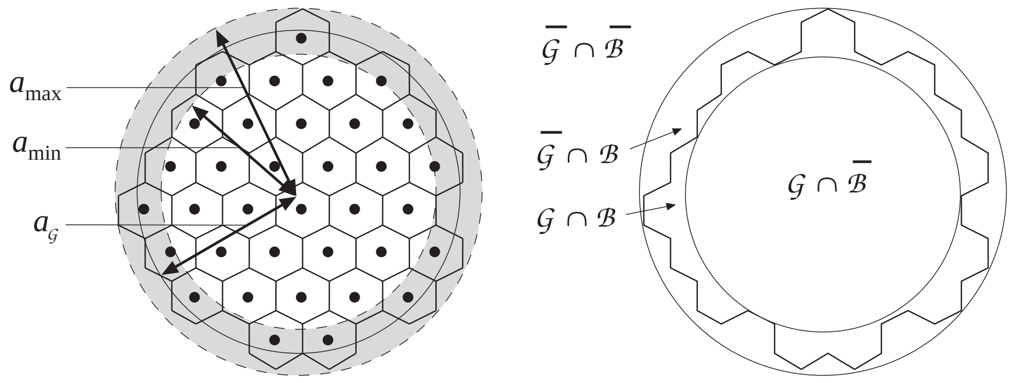

$a_{\max}$
$a_{\min}$
$a_{\mathcal{G}}$
$\overline{\mathcal{G}} \cap \overline{\mathcal{B}}$
$\overline{\mathcal{G}} \cap \mathcal{B}$
$\mathcal{G} \cap \mathcal{B}$
$\mathcal{G} \cap \overline{\mathcal{B}}$

Figure A.1. Left: Illustration of the border region (the gray area). Right: Combinations of the granular and the border region.

which overlaps both the granular and the overload region. The border shell is defined as the thinnest shell having only granular region on the inside and only overload region on the outside, that is, $a_{\min}$ is the radius of the inscribed sphere, and $a_{\max}$ is the radius of the circumscribed sphere of the granular region,

$$a_{\min} = \inf_{\mathbf{x} \in \overline{\mathcal{G}}} \|\mathbf{x}\| \tag{A.6}$$

$$a_{\max} = \sup_{\mathbf{x} \in \mathcal{G}} \|\mathbf{x}\|. \tag{A.7}$$

The border region is a mix of granular and overload regions. Figure A.1 illustrates the border region for a two-dimensional lattice VQ.

With the definitions of overload and granular regions in (3.11), and the border region in (A.5), we have

$$\overline{\mathcal{G}} \cap \overline{\mathcal{B}} = \left\{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| > a_{\max} \right\} \subseteq \overline{\mathcal{G}} \subseteq \left\{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| > a_{\min} \right\} = \overline{\mathcal{G}} \cup \mathcal{B} \tag{A.8}$$

$$\mathcal{G} \cap \overline{\mathcal{B}} = \left\{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \le a_{\min} \right\} \subseteq \mathcal{G} \subseteq \left\{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \le a_{\max} \right\} = \mathcal{G} \cup \mathcal{B}\,. \tag{A.9}$$

From (A.9), we conclude that the radius of the granular sphere, $a_{\mathcal{G}}$, is bounded between $a_{\min}$ and $a_{\max}$, since

$$\operatorname{vol}\left(S_d\left(a_{\min}\right)\right) \le \operatorname{vol}\left(\mathcal{G}\right) = \operatorname{vol}\left(S_d\left(a_{\mathcal{G}}\right)\right) \le \operatorname{vol}\left(S_d\left(a_{\max}\right)\right) \;\Rightarrow a_{\min} \le a_{\mathcal{G}} \le a_{\max}\,. \tag{A.10}$$

We use the covering radius $r_{\max}$, the packing radius $r_{\min}$, and the effective radius r_{Ω} of a granular Voronoi region $\Omega_{\mathcal{G}}(\mathbf{c})$, defined as

$$r_{\max} = \sup_{\mathbf{x} \in \Omega_{\mathcal{G}}(\mathbf{c})} \|\mathbf{x} - \mathbf{c}\| = \sup_{\mathbf{x} \in \Omega} \|\mathbf{x}\| \tag{A.11}$$

$$r_{\min} = \inf_{\mathbf{x} \notin \Omega_{\mathcal{G}}(\mathbf{c})} \|\mathbf{x} - \mathbf{c}\| = \inf_{\mathbf{x} \notin \Omega} \|\mathbf{x}\| \tag{A.12}$$

$$r_{\Omega} = \left(\frac{\operatorname{vol}(\Omega)}{\operatorname{vol}\left(S_d(1)\right)} \right)^{1/d} = \left(\frac{\operatorname{vol}\left(\mathcal{G}\right)}{N \cdot \operatorname{vol}\left(S_d(1)\right)} \right)^{1/d} = \left(\frac{\operatorname{vol}\left(S_d\left(a_{\mathcal{G}}\right)\right)}{N \cdot \operatorname{vol}\left(S_d(1)\right)} \right)^{1/d} = a_{\mathcal{G}} \cdot 2^{-R}. \tag{A.13}$$

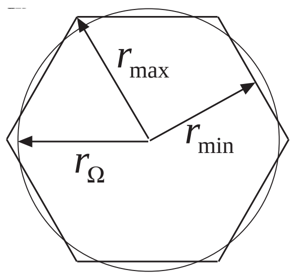


Figure A.2. A Voronoi region.

The three radii, r_Ω, $r_{\min}$ and $r_{\max}$, are illustrated in figure A.2. The granular Voronoi regions are all bounded and congruent, and thus the ratios $r_{\max}/r_\Omega$ and $r_{\min}/r_\Omega$ are bounded, nonzero and independent of the scaling of the region, so that

$$r_{\max} = \frac{r_{\max}}{r_\Omega} \cdot r_\Omega = \frac{r_{\max}}{r_\Omega} \cdot a_{\mathcal{G}} \cdot 2^{-R} \tag{A.14}$$

$$r_{\min} = \frac{r_{\min}}{r_\Omega} \cdot r_\Omega = \frac{r_{\min}}{r_\Omega} \cdot a_{\mathcal{G}} \cdot 2^{-R}. \tag{A.15}$$

$a_{\min}$ and $a_{\max}$ can be bounded as (using the definition of Λ in (3.1))

$$\begin{aligned} a_{\min} &= \inf_{\mathbf{x}\in\mathcal{G}} \|\mathbf{x}\| = \inf_{\mathbf{x}\in \bigcup_{\mathbf{c}_i\in\Lambda\setminus\mathcal{C}}(\Omega+\mathbf{c}_i)} \|\mathbf{x}\| = \inf_{\mathbf{c}_i\in\Lambda\setminus\mathcal{C}} \inf_{\mathbf{x}\in(\Omega+\mathbf{c}_i)} \|\mathbf{x}\| = \inf_{\mathbf{c}_i\in\Lambda\setminus\mathcal{C}} \inf_{\mathbf{x}\in\Omega} \|\mathbf{x}-\mathbf{c}_i\| \geq \\ &\geq \inf_{\mathbf{c}_i\in\Lambda\setminus\mathcal{C}} \inf_{\mathbf{x}\in\Omega} \left(\|\mathbf{c}_i\| - \|\mathbf{x}\|\right) = \inf_{\mathbf{c}_i\in\Lambda\setminus\mathcal{C}} \|\mathbf{c}_i\| - \sup_{\mathbf{x}\in\Omega} \|\mathbf{x}\| \geq a_{\mathrm{T}} - r_{\max}, \end{aligned} \tag{A.16}$$

$$\begin{aligned} a_{\max} &= \sup_{\mathbf{x}\in\mathcal{G}} \|\mathbf{x}\| = \sup_{\mathbf{x}\in \bigcup_{\mathbf{c}_i\in\mathcal{C}}(\Omega+\mathbf{c}_i)} \|\mathbf{x}\| = \sup_{\mathbf{c}_i\in\mathcal{C}} \sup_{\mathbf{x}\in(\Omega+\mathbf{c}_i)} \|\mathbf{x}\| = \sup_{\mathbf{c}_i\in\mathcal{C}} \sup_{\mathbf{x}\in\Omega} \|\mathbf{x}-\mathbf{c}_i\| \leq \\ &\leq \sup_{\mathbf{c}_i\in\mathcal{C}} \sup_{\mathbf{x}\in\Omega} \left(\|\mathbf{x}\| + \|\mathbf{c}_i\|\right) = \sup_{\mathbf{c}_i\in\mathcal{C}} \|\mathbf{c}_i\| + \sup_{\mathbf{x}\in\Omega} \|\mathbf{x}\| \leq a_{\mathrm{T}} + r_{\max}. \end{aligned} \tag{A.17}$$

where the last inequality of (A.16) and (A.17) follows from the truncation of the lattice by a hypersphere with radius a_{T}, as in (3.10). Now, using (A.10), (A.16) and (A.17), we can bound the truncation radius a_{T} as

$$a_{\mathcal{G}} - r_{\max} \leq a_{\max} - r_{\max} \leq a_{\mathrm{T}} \leq a_{\min} + r_{\max} \leq a_{\mathcal{G}} + r_{\max}. \tag{A.18}$$

The left- and right-most terms of (A.18) can both be written[11] $a_{\mathcal{G}} + r_{\max} \cdot O(1)$, and using (A.14), we get

$$a_{\mathrm{T}} = a_{\mathcal{G}} + r_{\max} \cdot O(1) = a_{\mathcal{G}} \cdot \left(1 + 2^{-R} \cdot O(1)\right). \tag{A.19}$$

Using (A.19) to eliminate $a_{\mathcal{G}}$ from (A.13)–(A.15), we get the useful equalities

[11] With $g \cdot O(1)$ (big-oh), we will mean $g \cdot C$, where C is bounded in a neighborhood of $g = 0$. Rules for computation using big-oh can be found in most mathematical handbooks, e.g. [49].

$$r_{\max} = a_{\mathrm{T}} \cdot 2^{-R} \cdot O(1) \tag{A.20}$$
$$r_{\Omega} = a_{\mathrm{T}} \cdot 2^{-R} \cdot O(1) \tag{A.21}$$
$$r_{\min} = a_{\mathrm{T}} \cdot 2^{-R} \cdot O(1), \tag{A.22}$$

and by inserting (A.20) into (A.18), we get

$$a_{\min} = a_{\mathrm{T}} \cdot \left(1 + 2^{-R} \cdot O(1)\right) \tag{A.23}$$
$$a_{\mathcal{G}} = a_{\mathrm{T}} \cdot \left(1 + 2^{-R} \cdot O(1)\right) \tag{A.24}$$
$$a_{\max} = a_{\mathrm{T}} \cdot \left(1 + 2^{-R} \cdot O(1)\right), \tag{A.25}$$

which illustrates that a_{T}, $a_{\mathcal{G}}$, $a_{\min}$, and $a_{\max}$ are all equal for infinite rates.

A.2 Theorem I: Overload distortion

In theorem I in section 3.2, we stated that the overload distortion is given by

$$D_{\overline{\mathcal{G}}} = f_{\overline{\mathcal{G}}}(d) \cdot a_{\mathrm{T}}^{d-4} \cdot e^{-a_{\mathrm{T}}^2/2} \cdot \left(1 + \varepsilon_{\overline{\mathcal{G}}}\right), \tag{A.26}$$

where $f_{\overline{\mathcal{G}}}(d) = \left(2^{d/2-2} \cdot \Gamma(d/2)\right)^{-1}$, and $\varepsilon_{\overline{\mathcal{G}}}$ tends to zero for asymptotically high rates R. In this section, we present a proof of this theorem. In the proof, we bound the overload distortion by use of two spheres, one outside and one inside the border region. Then we complete the proof by showing that the width of the border region tends to zero when the rate approaches infinity.

We write the overload distortion

$$D_{\overline{\mathcal{G}}} = \int_{\overline{\mathcal{G}}} \|\mathbf{x} - \mathbf{c}^*\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}, \tag{A.27}$$

where $\mathbf{c}^*$ is the codevector in the codebook $\mathcal{C}$ that is closest to the input vector $\mathbf{x}$, and $f_{\mathbf{x}}(\mathbf{x})$ is the input pdf. The integrand is nonnegative, so we can lower- and upper-bound the distortion by integrating over a smaller and larger region, respectively. Using (A.8), we get

$$\int_{\|\mathbf{x}\| > a_{\max}} \|\mathbf{x} - \mathbf{c}^*\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \le D_{\overline{\mathcal{G}}} \le \int_{\|\mathbf{x}\| > a_{\min}} \|\mathbf{x} - \mathbf{c}^*\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}. \tag{A.28}$$

We now study the upper and lower bound in (A.28) separately. First, noting that all codevectors lie inside a sphere with radius $a_{\max}$, we can lower-bound the integrand

$$\|\mathbf{x} - \mathbf{c}^*\| = \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\| \ge \min_{\|\mathbf{y}\| < a_{\max}} \|\mathbf{x} - \mathbf{y}\| = \|\mathbf{x} - \mathbf{p}(a_{\max})\|. \tag{A.29}$$

Secondly, the integrand can be upper-bounded by use of the triangle inequality,

$$\|\mathbf{x} - \mathbf{c}^*\| \le \|\mathbf{x} - \mathbf{p}(a_{\min})\| + \|\mathbf{p}(a_{\min}) - \mathbf{c}^*\|. \tag{A.30}$$

With the definition of $a_{\min}$ in (A.6), $\mathbf{p}(a_{\min})$ belongs to a granular Voronoi region. Therefore, we can bound the distance between $\mathbf{p}(a_{\min})$ and $\mathbf{c}^*$ by the covering radius of the Voronoi region, $r_{\max}$,

$$\left\| \mathbf{p}(a_{\min}) - \mathbf{c}^* \right\| \le r_{\max} \tag{A.31}$$

(see (A.11) and figure A.2). Thus, we have

$$\left\| \mathbf{x} - \mathbf{c}^* \right\| \le \| \mathbf{x} \| - a_{\min} + r_{\max} = \left\| \mathbf{x} - \mathbf{p}(a_{\min} - r_{\max}) \right\| \quad \text{if } \| \mathbf{x} \| > a_{\min}, \tag{A.32}$$

where we have also used (A.3). The distortion upper bound is

$$D_{\overline{\mathcal{G}}} \le \int\limits_{\|\mathbf{x}\|>a_{\min}} \left\| \mathbf{x} - \mathbf{p}(a_{\min} - r_{\max}) \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \le \int\limits_{\|\mathbf{x}\|>a_{\min}-r_{\max}} \left\| \mathbf{x} - \mathbf{p}(a_{\min} - r_{\max}) \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}. \tag{A.33}$$

Combining (A.28), (A.29) and (A.33), we get

$$\int\limits_{\|\mathbf{x}\|>a_{\max}} \left\| \mathbf{x} - \mathbf{p}(a_{\max}) \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \le D_{\overline{\mathcal{G}}} \le \int\limits_{\|\mathbf{x}\|>a_{\min}-r_{\max}} \left\| \mathbf{x} - \mathbf{p}(a_{\min} - r_{\max}) \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}, \tag{A.34}$$

which bounds the overload distortion by use of two spheres with radii $a_{\max}$ and $a_{\min} - r_{\max}$. From (A.20), (A.23) and (A.25), we see that both radii can be written on the same form, $a_{\mathrm{T}} \cdot \left(1 + 2^{-R} \cdot O(1)\right)$. We define

$$\hat{a} = a_{\mathrm{T}} \cdot \left(1 + 2^{-R} \cdot O(1)\right), \tag{A.35}$$

and rewrite the overload distortion as

$$D_{\overline{\mathcal{G}}} = \int\limits_{\|\mathbf{x}\|>\hat{a}} \left\| \mathbf{x} - \mathbf{p}(\hat{a}) \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \tag{A.36}$$

$$= \int\limits_{\|\mathbf{x}\|>\hat{a}} (\|\mathbf{x}\| - \hat{a})^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \tag{A.37}$$

$$= \int\limits_{\|\mathbf{x}\|>\hat{a}} \left(\| \mathbf{x} \|^2 + \hat{a}^2 - 2\hat{a} \| \mathbf{x} \| \right) f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}. \tag{A.38}$$

Now the *d*-dimensional integral has become one-dimensional; the integrand is a function of $\|\mathbf{x}\|$ only[12]. The stochastic variable $\xi = \| \mathbf{x} \|^2$ has a χ^2-distribution with *d* degrees of freedom, $f_\xi(\xi) = \chi^2(d, \xi)$, and we get

$$D_{\overline{\mathcal{G}}} = \int_{\hat{a}^2}^{\infty} \left(\xi + \hat{a}^2 - 2\hat{a}\sqrt{\xi} \right) \chi^2(d, \xi) d\xi \tag{A.39}$$

$$= \int_{\hat{a}^2}^{\infty} \left(\xi + \hat{a}^2 - 2\hat{a}\sqrt{\xi} \right) \frac{\xi^{d/2-1} \cdot e^{-\xi/2}}{2^{d/2} \cdot \Gamma(d/2)} d\xi. \tag{A.40}$$

In the sequel, we need the incomplete Gamma function,

[12] Since the Gaussian pdf $f_{\mathbf{x}}(\mathbf{x})$ is spherically symmetrical, it is a function of $\|\mathbf{x}\|$ only.

$$\Gamma(b,z)=\int_z^\infty t^{b-1}e^{-t}dt. \tag{A.41}$$

Using $\Gamma(b,z)$, we write the overload distortion as

$$D_{\overline{\mathcal{G}}}=\frac{1}{\Gamma(d/2)}\cdot\left[2\cdot\Gamma\left(\frac{d+2}{2},\frac{\hat{a}^2}{2}\right)-2\sqrt{2}\cdot\hat{a}\cdot\Gamma\left(\frac{d+1}{2},\frac{\hat{a}^2}{2}\right)+\hat{a}^2\cdot\Gamma\left(\frac{d}{2},\frac{\hat{a}^2}{2}\right)\right]. \tag{A.42}$$

We approximate the incomplete Gamma function as an asymptotic series [50]:

$$\Gamma(b,z)=z^{b-1}e^{-z}\left[1+(b-1)z^{-1}+(b-1)(b-2)z^{-2}+z^{-3}O(1)\right]. \tag{A.43}$$

With this approximation, the overload distortion can, after some work, be written

$$D_{\overline{\mathcal{G}}}=\frac{\hat{a}^{d-4}e^{-\hat{a}^2/2}}{2^{d/2-2}\cdot\Gamma(d/2)}\cdot\left(1+\hat{a}^{-2}\cdot O(1)\right) \tag{A.44}$$

Insertion of (A.35) yields, again omitting the details,

$$D_{\overline{\mathcal{G}}}=\frac{a_{\mathrm{T}}^{d-4}\cdot e^{-a_{\mathrm{T}}^2/2}}{2^{d/2-2}\cdot\Gamma(d/2)}\cdot\left[1+a_{\mathrm{T}}^2\cdot 2^{-R}\cdot O(1)+a_{\mathrm{T}}^{-2}\cdot O(1)\right], \tag{A.45}$$

which is equal to (A.26), and the proof is completed.

In section A.4, we will verify that the error term is equal to zero for asymptotically high rates if the truncation radius is selected for minimum distortion.

A.3 Theorem II: Granular distortion

In theorem II, the granular distortion is given by

$$D_{\mathcal{G}}=f_{\mathcal{G}}(d)\cdot 2^{-2R}\cdot a_{\mathrm{T}}^2\cdot\left(1+\varepsilon_{\mathcal{G}}\right) \tag{A.46}$$

where $f_{\mathcal{G}}(d)=G\cdot d\cdot\pi\cdot\Gamma(d/2+1)^{-2/d}$, and $\varepsilon_{\mathcal{G}}$ tends to zero for asymptotically high rates R. The proof of the theorem, which is given in this section, is based on writing the pdf inside each Voronoi region as a uniform pdf plus an error term. The granular distortion for a uniform pdf is easily computed, and the proof is completed by showing that the error term is zero for infinite rates.

We write the granular distortion for the N-point lattice VQ as a sum of the Voronoi region distortions

$$D_{\mathcal{G}}=\int_{\mathcal{G}}\left\|\mathbf{x}-\mathbf{c}^*\right\|f_{\mathbf{x}}(\mathbf{x})d\mathbf{x} \tag{A.47}$$

$$=\sum_{k=1}^{N}\int_{\Omega_{\mathcal{G}}(\mathbf{c}_k)}\left\|\mathbf{x}-\mathbf{c}_k\right\|f_{\mathbf{x}}(\mathbf{x})d\mathbf{x}. \tag{A.48}$$

For bounded and differentiable densities, we can expand the pdf in a Taylor series as

$$f_{\mathbf{x}}(\mathbf{x})=f_{\mathbf{x}}(\mathbf{c}_k)+\left\|\mathbf{x}-\mathbf{c}_k\right\|\cdot O(1), \tag{A.49}$$

and (A.48) can be rewritten as

$$D_{\mathcal{G}} = \sum_{k=1}^{N} \left(\int_{\Omega_{\mathcal{G}}(\mathbf{c}_k)} \|\mathbf{x}-\mathbf{c}_k\|^2 \left(f_{\mathbf{x}}(\mathbf{c}_k) + \|\mathbf{x}-\mathbf{c}_k\| \cdot O(1) \right) d\mathbf{x} \right) \tag{A.50}$$

$$= \sum_{k=1}^{N} \left(f_{\mathbf{x}}(\mathbf{c}_k) \cdot \int_{\Omega_{\mathcal{G}}(\mathbf{c}_k)} \|\mathbf{x}-\mathbf{c}_k\|^2 d\mathbf{x} \right) + \sum_{k=1}^{N} \left(\int_{\Omega_{\mathcal{G}}(\mathbf{c}_k)} \|\mathbf{x}-\mathbf{c}_k\|^3 d\mathbf{x} \cdot O(1) \right). \tag{A.51}$$

Now, since the granular Voronoi regions $\Omega_{\mathcal{G}}(\mathbf{c}_k)$ are congruent, the integrals in (A.51) are independent of *k*, and we get

$$D_{\mathcal{G}} = \int_{\Omega} \|\mathbf{x}\|^2 d\mathbf{x} \cdot \sum_{k=1}^{N} f_{\mathbf{x}}(\mathbf{c}_k) + \sum_{k=1}^{N} \int_{\Omega} \|\mathbf{x}\|^3 d\mathbf{x} \cdot O(1). \tag{A.52}$$

The first integral in (A.52) is recognized to be a scaled version of the lattice quantization constant *G* (3.6). The second integral can be simplified by using (A.11), and writing $\|\mathbf{x}\| = r_{\max} \cdot O(1)$. We get

$$D_{\mathcal{G}} = d \cdot \mathrm{vol}(\Omega)^{1+2/d} \cdot G \cdot \sum_{k=1}^{N} f_{\mathbf{x}}(\mathbf{c}_k) + r_{\max}^3 \cdot \mathrm{vol}(\mathcal{G}) \cdot O(1) \tag{A.53}$$

$$= d \cdot \mathrm{vol}(\Omega)^{1+2/d} \cdot G \cdot \sum_{k=1}^{N} f_{\mathbf{x}}(\mathbf{c}_k) + a_{\mathrm{T}}^{d+3} \cdot 2^{-3R} \cdot O(1), \tag{A.54}$$

where (A.20) is used for the last equality.

The sum in (A.54) is considered next. For this reason, we study the granular probability $\Pr(\mathbf{x} \in \mathcal{G})$. Using the same approach as in (A.47)–(A.54), we can write the granular probability

$$\Pr(\mathbf{x} \in \mathcal{G}) = \int_{\mathcal{G}} f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \tag{A.55}$$

$$= \mathrm{vol}(\Omega) \cdot \sum_{k=1}^{N} f_{\mathbf{x}}(\mathbf{c}_k) + a_{\mathrm{T}}^{d+1} \cdot 2^{-R} \cdot O(1). \tag{A.56}$$

We can also write the granular probability using the overload probability, as

$$\Pr(\mathbf{x} \in \mathcal{G}) = 1 - \Pr(\mathbf{x} \in \overline{\mathcal{G}}). \tag{A.57}$$

Using (A.8), we can bound the overload probability as

$$\Pr(\mathbf{x} \in \overline{\mathcal{G}}) \le \Pr(\|\mathbf{x}\| > a_{\min}). \tag{A.58}$$

(A.58) can be written using the χ^2-distribution as in (A.39). We get

$$\Pr(\mathbf{x} \in \overline{\mathcal{G}}) \le \frac{\Gamma(d/2, a_{\min}^2/2)}{\Gamma(d/2)}, \tag{A.59}$$

which can be simplified using the first term in (A.43),

$$\Pr\left(\mathbf{x} \in \overline{\mathcal{G}}\right) = a_{\min}^{d-2} \cdot e^{-a_{\min}^2/2} \cdot O(1) = a_{\mathrm{T}}^{d-2} \cdot e^{-a_{\mathrm{T}}^2/2} \cdot O(1) \qquad \text{(A.60)}$$

(see (A.23)). Combining (A.56), (A.57) and (A.60), we get

$$\mathrm{vol}(\Omega) \cdot \sum_{k=1}^{N} f_{\mathbf{x}}\left(\mathbf{c}_k\right) = 1 + a_{\mathrm{T}}^{d+1} \cdot 2^{-R} \cdot O(1) + a_{\mathrm{T}}^{d-2} \cdot e^{-a_{\mathrm{T}}^2/2} \cdot O(1). \qquad \text{(A.61)}$$

Using the number of codevectors in the quantizer, $N = 2^{R \cdot d}$, the volume of the Voronoi region, $\mathrm{vol}(\Omega)$, can be written

$$\mathrm{vol}(\Omega) = \frac{\mathrm{vol}(\mathcal{G})}{N} = \frac{\mathrm{vol}\left(S_d\left(a_{\mathcal{G}}\right)\right)}{N} = \frac{\pi^{d/2} \cdot a_{\mathcal{G}}^{d} \cdot 2^{-Rd}}{\Gamma(d/2+1)}, \qquad \text{(A.62)}$$

where we have used the fact that the volume of the granular region, $\mathrm{vol}(\mathcal{G})$, is equal to the volume of a *d*-sphere [50] with radius $a_{\mathcal{G}}$, see (A.10). Inserting (A.24), the volume of the lattice Voronoi region is expressed as a function of the truncation radius a_{T},

$$\mathrm{vol}(\Omega) = \frac{\pi^{d/2} \cdot a_{\mathrm{T}}^{d} \cdot 2^{-Rd}}{\Gamma(d/2+1)} \cdot \left(1 + 2^{-R} \cdot O(1)\right). \qquad \text{(A.63)}$$

Inserting (A.61) and (A.63) into (A.54), we get

$$D_{\mathcal{G}} = \frac{G \cdot d \cdot \pi}{\Gamma^{2/d}(d/2+1)} \cdot a_{\mathrm{T}}^{2} \cdot 2^{-2R} \cdot \left[1 + a_{\mathrm{T}}^{d+1} \cdot 2^{-R} \cdot O(1) + a_{\mathrm{T}}^{d-2} \cdot e^{-a_{\mathrm{T}}^2/2} \cdot O(1)\right], \text{(A.64)}$$

which equals (A.46). If the error terms in (A.64) are excluded, the equation describes the distortion for quantization of a spherical uniform pdf (see [14], (1.10)).

In section A.4, we show that for an optimal choice of a_{T}, the error terms tend to zero for a rate approaching infinity.

A.4 Total distortion

The key issue in the high rate theory is to find the optimal value of the truncation radius a_{T}. We study three possible choices of a_{T}:

I a_{T} does not grow towards infinity with the rate.

II a_{T} grows towards infinity with the rate, but slower than exponentially in *R*.

III a_{T} grows towards infinity exponentially in *R*, or even faster, i.e. $a_{\mathrm{T}} \geq 2^{\lambda R}$ for some λ.

We show in the following that I and III lead to higher distortion than II. For this reason, we use an arbitrary formula for a_{T} fulfilling II, and compute the resulting distortion. Then we lower-bound the distortion in I and III by simple calculations. The proof is completed by showing that the distortion for case II is lower than the lower bounds of distortion for case I and III.

First we study the distortion for case II above. For this case, the error terms in (A.26), (A.45) and (A.46), (A.64) are zero for asymptotically high rates. The total distortion is the sum of (A.26) and (A.46),

$$D = \left(f_{\mathcal{G}} \cdot a_{\mathrm{T}}^2 \cdot 2^{-2R} + f_{\overline{\mathcal{G}}} \cdot a_{\mathrm{T}}^{d-4} \cdot e^{-a_{\mathrm{T}}^2/2} \right)(1+\varepsilon). \tag{A.65}$$

We select the truncation radius arbitrarily as $a_{\mathrm{T}} = R$, which fulfills II. Insertion of a_{T} in (A.65) yields

$$D_{\mathrm{II}} = \left(f_{\mathcal{G}} \cdot R^2 \cdot 2^{-2R} + f_{\overline{\mathcal{G}}} \cdot R^{d-4} \cdot e^{-R^2/2} \right)(1+\varepsilon) \tag{A.66}$$

$$= f_{\mathcal{G}} \cdot R^2 \cdot 2^{-2R} \cdot \left(1 + R^{d-4} \cdot e^{-R^2/2} \cdot 2^{2R} \cdot O(1) \right)(1+\varepsilon) \tag{A.67}$$

$$= f_{\mathcal{G}} \cdot R^2 \cdot 2^{-2R} \cdot \left(1 + R^{d-4} \cdot e^{-R^2/2 + R\cdot 2\cdot \ln 2} \cdot O(1) \right)(1+\varepsilon). \tag{A.68}$$

The error terms are zero for infinite rates. We write

$$D_{\mathrm{II}} = R^2 \cdot 2^{-2R} \cdot O(1), \tag{A.69}$$

and observe that the distortion tends to zero when the rate approaches infinity. Since we have used an arbitrary truncation radius fulfilling II, the optimal truncation radius gives a distortion lower than or equal to (A.69).

Now we study case I. In (A.34) a lower bound for the overload distortion is given. Using (A.17) we get

$$D_{\mathrm{I}} \geq D_{\overline{\mathcal{G}}} \geq \int_{\|\mathbf{x}\| > a_{\max}} \left\| \mathbf{x} - \mathbf{p}(a_{\max}) \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \geq \int_{\|\mathbf{x}\| > a_{\mathrm{T}} + r_{\max}} \left\| \mathbf{x} - \mathbf{p}(a_{\max}) \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x}. \tag{A.70}$$

We observe that, for finite a_{T} and $r_{\max}$, the right-hand integral in (A.70) does not tend to zero as the rate approaches infinity. Since a_{T} is finite in case I, and $r_{\max}$ is finite for finite a_{T} (A.20), we conclude that D_{I} does not tend to zero as the rate approaches infinity. But $D_{\mathrm{II}} \to 0$ for $R \to \infty$, and we have shown that the optimal high-rate distortion in case I is higher than the distortion in case II, i.e. $D_{\mathrm{I}} > D_{\mathrm{II}}$.

To lower-bound the distortion in case III, we first define a shape $\mathcal{S}$ in the form of a *d*-sphere from which we cut out spherical holes around all codevectors **c**,

$$\mathcal{S} = S_d(\beta) \setminus \bigcup_{\mathbf{c} \in C} \left(S_d(\alpha \cdot r_{\min}) + \mathbf{c} \right), \tag{A.71}$$

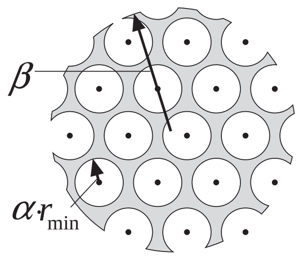


Figure A.3. The hollow shape $\mathcal{S}$.

where $0 < \alpha < 1$, and the radius β is an arbitrary constant, independent of *R*. $\mathcal{S}$ is illustrated in figure A.3. Since α is less than 1, the definition of $r_{\min}$ (A.12) ensures that the holes, with radius $\alpha \cdot r_{\min}$, are nonoverlapping. Further, since the truncation radius a_{T} (and $a_{\min}$, see (A.23)) grows towards infinity with the rate, there exists a constant R_0 such that for all rates $R > R_0$, $a_{\min} > \beta$, which makes $\mathcal{S}$ a subset of the granular region $\mathcal{G}$. We have

$$D_{\mathrm{III}} \geq D_{\mathcal{G}} = \int_{\mathcal{G}} \left\| \mathbf{x} - \mathbf{c}^* \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \geq \int_{S} \left\| \mathbf{x} - \mathbf{c}^* \right\|^2 f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} \quad \text{for } R > R_0 . \qquad \text{(A.72)}$$

For vectors $\mathbf{x}$ in $\mathcal{S}$, the distance to the closest codeword $\mathbf{c}^*$ is lower-bounded by $\alpha \cdot r_{\min}$. The pdf $f_{\mathbf{x}}(\mathbf{x})$ is lower-bounded by the pdf at an arbitrary point at the surface of $\mathcal{S}$, i.e. $f_{\mathbf{x}}(\mathbf{x}) \leq f_{\mathbf{x}}\left(\mathbf{x}_\beta\right)$ where $\left\|\mathbf{x}_\beta\right\| = \beta$. Thus, for $R > R_0$, we have that (using (A.18))

$$D_{\mathrm{III}} \geq \int_{S} \left(\alpha \cdot r_{\min}\right)^2 f_{\mathbf{x}}\left(\mathbf{x}_\beta\right) d\mathbf{x} \qquad \text{(A.73)}$$

$$= \left(\alpha \cdot r_{\min}\right)^2 \cdot f_{\mathbf{x}}\left(\mathbf{x}_\beta\right) \cdot \mathrm{vol}\left(\mathcal{S}\right) \qquad \text{(A.74)}$$

$$= f_{\mathbf{x}}\left(\mathbf{x}_\beta\right) \cdot \mathrm{vol}\left(\mathcal{S}\right) \cdot \left(\alpha \cdot r_{\min} / r_{\Omega}\right)^2 \cdot a_{\mathcal{G}}^2 \cdot 2^{-2R} \qquad \text{(A.75)}$$

$$\geq f_{\mathbf{x}}\left(\mathbf{x}_\beta\right) \cdot \mathrm{vol}\left(\mathcal{S}\right) \cdot \left(\alpha \cdot r_{\min} / r_{\Omega}\right)^2 \cdot \left(a_{\mathrm{T}} - r_{\max}\right)^2 \cdot 2^{-2R} \qquad \text{(A.76)}$$

$$\geq C \cdot a_{\mathrm{T}}^2 \cdot 2^{-2R}, \qquad \text{(A.77)}$$

where *C* is a positive constant, since $r_{\max} / a_{\mathrm{T}}$ tends to zero (see (A.20)), and the volume of $\mathcal{S}$, the pdf $f_{\mathbf{x}}\left(\mathbf{x}_\beta\right)$ at the surface, and $r_{\min} / r_{\Omega}$ are all positive constants. Now, inserting a_{T} as in case III yields

$$D_{\mathrm{III}} \geq C \cdot 2^{2\lambda R} \cdot 2^{-2R} > D_{\mathrm{II}} \text{ for R} \to \infty, \qquad \text{(A.78)}$$

and we have shown that radius selection as in case II leads to lower distortion than case III.

We will now study the total distortion, *D*, and show that, for a selection of a_{T} with the restrictions as in case II above, the distortion is convex and has a distinct global minimum. As discussed above, the error term in (A.65) is zero for infinite rate. We define $\hat{D}$ as *D* excluding the error term,

$$\hat{D} = f_{\mathcal{G}} \cdot a_{\mathrm{T}}^2 \cdot 2^{-2R} + f_{\bar{\mathcal{G}}} \cdot a_{\mathrm{T}}^{d-4} \cdot e^{-a_{\mathrm{T}}^2/2} . \qquad \text{(A.79)}$$

To show that $\hat{D}$ is convex with respect to a_{T}, we compute the second derivative of $\hat{D}$ with respect to a_{T}:

$$\frac{\partial^2 \hat{D}}{\partial a_{\mathrm{T}}^2} = 2 \cdot f_{\mathcal{G}} \cdot 2^{-2R} + f_{\bar{\mathcal{G}}} \cdot a_{\mathrm{T}}^{d-2} \cdot e^{-a_{\mathrm{T}}^2/2} \cdot \left[1 + (7 - 2d) \cdot a_{\mathrm{T}}^{-2} + \left(d^2 - 9d + 20\right) \cdot a_{\mathrm{T}}^{-4}\right]. \qquad \text{(A.80)}$$

We see that the expression inside brackets is dominated by the first term when a_{T} tends to infinity, and we write

$$\frac{\partial^2 \hat{D}}{\partial a_{\mathrm{T}}^2} = 2 \cdot f_{\mathcal{G}} \cdot 2^{-2R} + f_{\bar{\mathcal{G}}} \cdot a_{\mathrm{T}}^{d-2} \cdot e^{-a_{\mathrm{T}}^2/2} \cdot \left(1 + a_{\mathrm{T}}^{-2} \cdot O(1)\right). \qquad \text{(A.81)}$$

Clearly, this expression is positive for large enough values of a_{T}. Thus, $\hat{D}$ is a convex function of a_{T} in the region defined in case II, and the first derivative can only be zero at the global minimum of $\hat{D}$. D is the sum of $\hat{D}$ and error terms, but $\hat{D}$ dominates the distortion for all a_{T} satisfying case II, so the global minimum of $\hat{D}$ is the global minimum of D as well.

BIBLIOGRAPHY

[1] B.-H. Juang and A. H. Gray, Jr., “Multiple stage vector quantization for speech coding,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Paris, France, vol. 1, pp. 597–600, 1982.

[2] R. M. Gray and H. Abut, “Full search and tree searched vector quantization of speech waveforms,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Paris, France, pp. 593–596, 1982.

[3] Y. Linde, A. Buzo, and R. M. Gray, “An algorithm for vector quantizer design,” *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, January 1980.

[4] J. Makhoul, S. Roucos, and H. Gish, “Vector quantization in speech coding,” *Proceedings of the IEEE*, vol. 73, no. 11, pp. 1551–1588, November 1985.

[5] A. Gersho and V. Cuperman, “Vector quantization: A pattern-matching technique for speech coding,” *IEEE Communications Magazine*, vol. 21, no. 9, pp. 15–21, December 1983.

[6] I. A. Gerson and M. A. Jasiuk, “Vector sum excited linear prediction (VSELP) speech coding at 8 kbps,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Albuquerque, USA, pp. 461–464, 1990.

[7] M. J. Sabin and R. M. Gray, “Product code vector quantizers for speech and waveform coding,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, pp. 474–488, June 1984.

[8] J. D. Gibson and K. Sayood, “Lattice quantization,” *Advances in Electronics and Electron Physics*, vol. 72, pp. 259–330, 1988.

[9] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups*, Second ed., Springer-Verlag, 1992.

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

[11] E. Agrell and P. Hedelin, “How to evaluate search methods for vector quantization,” *in Proc. NORSIG*, Ålesund, Norway, pp. 258–263, 1994.

[12] V. Ramasubramanian and K. K. Paliwal, “An efficient approximation-elimination algorithm for fast nearest-neighbor search based on a spherical distance coordinate formulation,” *Pattern Recognition Letters*, vol. 13, no. 7, pp. 471–480, July 1992.

[13] 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, March 1992.

[14] E. Agrell and T. Eriksson, “Lattice-based quantization, part I,” Technical report no. 17, Chalmers University of Technology, October, 1996.

[15] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, nos. 3 and 4, pp. 379–423 and 623–656, July and October 1948.

[16] A. Gersho and R. M. Gray, *Vector quantization and signal compression*, Kluwer Academic Publishers, 1992.

[17] G. Voronoï, “Nouvelles applications des paramètres continus à la théorie des formes quadratiques,” *Journal für die reine und angewandte Mathematik*, vol. 133, pp. 97–178, 1908; vol. 134, pp. 198–287, 1908; and vol. 136, pp. 67–181, 1909.

[18] R. M. Gray, *Source coding theory*, Kluwer Academic Publishers, 1990.

[19] P. Knagenhjelm, *Competitive learning in robust communication*, PhD dissertation, Chalmers University of Technology, Gothenburg, Sweden, 1993.

[20] M. Antonini, M. Barlaud, and T. Gaidon, “Adaptive entropy constrained lattice vector quantization for multiresolution image coding,” *Proceedings of SPIE Visual Communications and Image processing*, vol. 1818, no. 2, pp. 441–457, November 1992.

[21] A. Woolf and G. Rogers, “Lattice vector quantization of image wavelet coefficient vectors using a simplified form of entropy coding,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Adelaide, Australia, vol. 5, pp. 269–272, 1994.

[22] J. Pan and T. R. Fischer, “Vector quantization-lattice vector quantization of speech LPC coefficients,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Adelaide, Australia, vol. 1, pp. 513–516, 1994.

[23] M. Xie and J.-P. Adoul, “Embedded algebraic vector quantizers (EAVQ) with application to wideband speech coding,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, USA, vol. 1, pp. 240–243, 1996.

[24] N. Moayeri and D. L. Neuhoff, “Theory of lattice-based fine-coarse vector quantization,” *IEEE Transactions on Information Theory*, vol. 37, no. 4, pp. 1072–1084, July 1991.

[25] N. Moayeri, D. L. Neuhoff, and W. E. Stark, “Fine-coarse vector quantization,” *IEEE Transactions on Signal Processing*, vol. 39, no. 7, pp. 1503–1515, July 1991.

[26] F. Kuhlmann and J. A. Bucklew, “Piecewise uniform vector quantizers,” *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1259–1263, September 1988.

[27] P. F. Swaszek, “Unrestricted multistage vector quantizers,” *IEEE Transactions on Information Theory*, vol. 38, no. 3, pp. 1169–1174, May 1992.

[28] T. Eriksson, “Dual-stage vector quantization with dynamic bit allocation,” *in Proc. EUSIPCO -94*, Edinburgh, Scotland, vol. 1, pp. 383–386, 1994.

[29] M. V. Eyuboğlu and G. D. Forney, Jr., “Lattice and trellis quantization with lattice- and trellis-bounded codebooks—High rate theory for memoryless sources,” *IEEE Transactions on Information Theory*, vol. 39, no. 1, pp. 46–59, January 1993.

[30] D. G. Jeong and J. D. Gibson, “Uniform and piecewise uniform lattice vector quantization for memoryless Gaussian and Laplacian sources,” *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 786–804, May 1993.

[31] T. R. Fischer and J. Pan, “Enumeration encoding and decoding algorithms for pyramid cubic lattice and trellis coding,” *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 2056–2061, November 1995.

[32] P. F. Swaszek, “A vector quantizer for the Laplace source,” *IEEE Transactions on Information Theory*, vol. 37, no. 5, pp. 1355–1365, September 1991.

[33] J. H. Conway and N. J. A. Sloane, “A fast encoding method for lattice codes and quantizers,” *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 820–824, November 1983.

[34] D. G. Jeong and J. D. Gibson, “Lattice vector quantization for image coding,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Glasgow, Scotland, pp. 1743–1746, 1989.

[35] G. D. Forney, Jr., “Multidimensional constellations—Part II: Voronoi constellations,” *IEEE Journal on Selected Areas in Communication*, vol. 7, pp. 941–958, Aug. 1989.

[36] S. P. Lloyd, “Least squares quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, March 1982.

[37] T. Kohonen, *Self-organizing and associative memory*, New York, Springer Verlag, 1984.

[38] K. Zeger, J. Vaisey, and A. Gersho, “Globally optimal vector quantizer design by stochastic relaxation,” *IEEE Transactions on Signal Processing*, vol. 40, no. 2, pp. 310–322, February 1992.

[39] N. Ueda and R. Nakano, “A new competitive learning approach based on an equidistortion principle for designing optimal quantizers,” *Neural Networks*, vol. 7, no. 8, pp. 1211–1227, 1994.

[40] P. Knagenhjelm, “A recursive design method for robust vector quantization,” *in Proc. International Conference on Signal Processing Applications and Technology*, Boston, pp. 948–954, 1992.

[41] J. A. Bucklew, “Companding and random quantization in several dimensions,” *IEEE Transactions on Information Theory*, vol. 27, no. 2, pp. 207–211, March 1981.

[42] S. Arya and D. M. Mount, “Algorithms for fast vector quantization,” *in Proc. Data Compression Conference*, Snowbird, USA, pp. 381–390, 1993.

[43] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, September 1975.

[44] D.-Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham, “Fast search algorithms for vector quantization and pattern matching,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, San Diego, USA, vol. 1, pp. 9.11.1–9.11.4, 1984.

[45] K. Motoishi and T. Misumi, “Fast vector quantization algorithm by using an adaptive search technique,” *in Proc. IEEE International Symposium on Information Theory*, San Diego, USA, pp. 76, 1990.

[46] J. Thyssen and S. D. Hansen, “Using neural networks for vector quantization in low rate speech coders,” *in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Minneapolis, USA, vol. 2, pp. 431–434, 1993.

[47] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C—the art of scientific computing*, Second ed., Cambridge University Press, 1992.

[48] 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, September 1977.

[49] D. E. Knuth, *The art of computer programming*, vol. 1, Second ed., Addison-Wesley Publishing Company, 1973.

[50] I. S. Gradshteyn and I. M. Ryzhik, *Table of integrals, series, and products*, 5th ed., San Diego, Academic Press, Inc., 1994.

ML Optimal CDMA Multiuser Receiver

Erik Agrell and Tony Ottosson

Published in
Electronics Letters,
vol. 31, no. 18, pp. 1554–1555, Aug. 1995.

Errors in the original version have been corrected as follows.

- p. 1554, col. 2, l. 8: $O(1.5^K)$ ➡ $O(1.5^K)$.
- p. 1554, col. 2, l. 10: transmits, at a given time with power w_k, ➡ transmits with power w_k, at a given time,
- p. 1554, col. 2, l. 13: w_k ➡ w_K
- p. 1554, col. 2, l. 33 from the bottom: with one ➡ one
- p. 1554, col. 2, l. 23 f.t.b.: ($K-1$)- ➡ (($K-1$-
- p. 1555, col. 1, l. 3: 3^K-2^K ➡ 3^K-2^K.
- p. 1555, col. 1, l. 14: neighbors [1] ➡ neighbors
- p. 1555, col. 1, l. 9 f.t.b.: detection is ➡ detection in
- p. 1555, col. 2, ll. 1–4: deleted
- 3 locations: parallellotope ➡ parallelepiped

The layout has been revised. Especially, fonts for variables have been more carefully chosen than in the journal.

Voronoi Regions for Binary Linear Block Codes

Erik Agrell

Published in
IEEE Transactions on Information Theory,
vol. 42, no. 1, pp. 310–316, Jan. 1996.

Errors in the original version have been corrected as follows.

p. 311, col. 1, l. 5 from the bottom: equality ➡ inequality

p. 313, col. 1, l. 2: Part ➡ part

p. 313, col. 2, l. 13 f.t.b.: $\frac{\sqrt{w\,SNR}}{2\sqrt{2}}$ ➡ $\sqrt{\frac{w\,SNR}{2}}$

7 locations: non-neighbors ➡ nonneighbors

The layout has been revised.

On the Complexity of Soft-Decision Decoding

Erik Agrell

Submitted to
IEEE Transactions on Information Theory,
Jan. 1997.

Errors in the submitted version have been corrected as follows.

p. 12, l. 9 from the bottom: observation ➡ observations
p. 17, l. 6 f.t.b.: E. Agrell ➡ E. Agrell and P. Hedelin
19 locations: soft decision ➡ soft-decision

The layout has been revised.

2	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	0	0
1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	1	0
1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0.573

ISBN 91-7197-464-4