Near-Ellipsoidal Lattice Quantization by Generalized Voronoi Shaping

Stéphane Ragot, Minjie Xie*, and Roch Lefebvre

Abstract— **This article introduces a class of index-optimized near-ellipsoidal lattice codes, which we refer to as "generalized Voronoi codes". We derive a necessary condition for the existence of such codes and present fast indexing algorithms. We also investigate several efficient strategies for outlier saturation. Algorithmic details are provided for the lattices A_2, D_n, and RE_{2n} only.**

Keywords— **Vector quantization, lattice, lattice indexing, nearest-neighbor search.**

I. Introduction

We address the problem of designing efficient fixed-rate lattice quantizers with ellipsoidal shaping. This work is mainly motivated by recent developments in speech coding, and specifically by spectrum coding with principal component analysis and Gaussian mixture modelling [6].

The lattice shaping problem was initially stated in the context of modulation to reduce the average signal power [10], and to minimize the average probability of error given dimension and average energy [11]. In the context of vector quantization this problem arises when optimizing a lattice truncation according to the source memory or the marginal density shape. We consider only the case of ellipsoidal shaping for vector quantization. This sub-problem was already studied in [7], [8], [9]. However, [7] used the lattice $\mathbb{Z}$ which provides no granular gain. And in [8] the indexing of codevectors was not considered, but rather addressed in [9].

In this paper, we present a generalization of Voronoi coding [1] which yields near-ellipsoidal lattice codes with fast indexing algorithms. We also propose several efficient strategies for outlier saturation. By lack of space, only the lattices A_2, D_n and RE_{2n} are considered herein, the proofs are omitted and the performance of the nearest-neighbor search algorithms is not presented.

II. Notations and definitions

Before proceeding further, we shall define the necessary conventions and notations. The scalar operators $\lfloor . \rfloor$ and $\lceil . \rceil$ round any input in $\mathbb{R}$ to the nearest integer in $\mathbb{Z}$ towards $-\infty$ and $+\infty$, respectively. The row convention is used for vectors. Therefore, if $\mathbf{x}$ is a vector in $\mathbb{R}^n$, and by denoting its transpose as $\mathbf{x}^T$, $\mathbf{x}\mathbf{x}^T$ corresponds to the squared Euclidean norm of $\mathbf{x}$. The vector operators `mod`, `mult` and `div` denote the element-by-element modulo, multiplication and division, respectively. In other words, if $\mathbf{x} = [x_1 \cdots x_n]$ is a vector in $\mathbb{R}^n$, $\mathbf{j} = [j_1 \cdots j_n]$ a vector in $\mathbb{Z}^n$ and $\mathbf{m} = [m_1 \cdots m_n]$ a vector in $\mathbb{N}^{*n}$ (i.e. a vector of positive integers with non-zero components), then

$$\mathbf{mod}(\mathbf{j}, \mathbf{m}) = [j_1 (mod\, m_1) \cdots j_n (mod\, m_n)] \quad (1)$$

$$\mathbf{mult}(\mathbf{x}, \mathbf{m}) = [x_1 m_1 \cdots x_n m_n] \quad (2)$$

$$\mathbf{div}(\mathbf{x}, \mathbf{m}) = [x_1 / m_1 \cdots x_n / m_n] \quad (3)$$

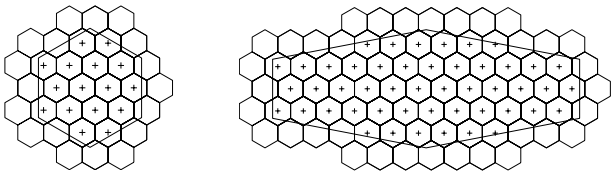
where *mod* is the scalar modulo operator defined as:

$$x\,(mod\, m) = x - m\lfloor x/m \rfloor = \begin{cases} x - m\lfloor x/m \rfloor, & x \geq 0 \\ x + m\lceil -x/m \rceil & x < 0 \end{cases}$$

In the general case, a lattice in $\mathbb{R}^n$ is denoted Λ, with generator matrix G_Λ. We use the row convention for G_Λ. That is, if $\mathbf{k}$ is a vector in $\mathbb{Z}^n$, $\mathbf{y} = \mathbf{k}G_\Lambda$ generates a point in Λ and $\mathbf{j} = \mathbf{y}G_\Lambda^{-1}$ retrieves the related basis expansion. We will use more specifically the lattices A_2, D_n and RE_{2n}. The lattices A_2 and D_n are defined in [13], and we define RE_{2n} as follows:

$$RE_{2n} = 2D_{2n} \cup \left\{ 2D_{2n} + (1^{2n}) \right\} \quad (4)$$

The Voronoi region related to the lattice point $\mathbf{y}$ in Λ is denoted $V_\Lambda(\mathbf{y})$. In a lattice all Voronoi regions are congruent, and we can consider $V_\Lambda(\mathbf{0})$ only. For a vector $\mathbf{m}$ in $\mathbb{N}^{*n}$, $V_\Lambda(\mathbf{0}, \mathbf{m})$ represents the region $V_\Lambda(\mathbf{0})$ scaled non-uniformly according to $\mathbf{m}$ (i.e. $V_\Lambda(\mathbf{0}, \mathbf{m}) = \mathbf{mult}(V_\Lambda(\mathbf{0}), \mathbf{m})$).



(a) Voronoi code (b) generalized Voronoi code

Fig. 1. Example of Voronoi codes ('+') derived from the A_2 lattice.

A. Voronoi coding

A Voronoi code of size 2^{nR}, R integer > 0, is defined as $C(\Lambda, R, \mathbf{a}) = \Lambda \cap \left(2^R V_\Lambda(\mathbf{0}) + \mathbf{a}\right)$, where $\mathbf{a}$ is an appropriate offset. Therefore $C(\Lambda, R, \mathbf{a})$ corresponds to a truncation of Λ by a scaled Voronoi region $2^R V_\Lambda(\mathbf{0})$ translated by $\mathbf{a}$, as illustrated in Fig. 1 (a). The extension of this definition to lattice shaping by geometrically-similar sublattices [12] is not considered herein.

Voronoi shaping yields index-optimized lattice codes since indices and codevectors in $C(\Lambda, R, \mathbf{a})$ can be generated efficiently from the generator matrix of Λ by lattice decoding and modular arithmetics.

This work was financed by the NSERC and VoiceAge Corp.

S. Ragot and R. Lefebvre are with the Department of Electrical and Computer Engineering, University of Sherbrooke, Sherbrooke, Québec, J1K 2R1 Canada. E-mail: {ragot,lefebvre}@gel.usherb.ca

*Work done when at the University of Sherbrooke.

Voronoi codes were introduced in [1] with fast indexing algorithms, but without considering nearest-neighbor search. The latter topic was addressed in [2]. In particular, it was suggested in [2] that an outlier can be projected on a properly scaled and translated Voronoi facet.

B. Generalized Voronoi coding

We define generalized Voronoi codes by extending the Voronoi lattice shaping of [1] to Voronoi regions with non-uniform component scaling, as depicted in Fig. 1 (b). Given $\mathbf{m}$ in $\mathbb{N}^{*n}$, we obtain $C(\Lambda, \mathbf{m}, a) = \Lambda \cap (V_\Lambda(\mathbf{0}, \mathbf{m}) + \mathbf{a})$. The resulting code size is $\prod_{i=1}^{n} m_i$. The offset $\mathbf{a}$ is still chosen to ensure that no lattice point belongs to the surface of the scaled and translated Voronoi region $(V_\Lambda(\mathbf{0}, \mathbf{m}) + \mathbf{a})$. As we shall see later, if we generalize the indexing algorithms described in [1] for these codes, constraints on $\mathbf{m}$ may apply [3].

The idea of generalized Voronoi coding was introduced in [3] – however, in [3] generalized Voronoi codes were defined for specific lattices (namely, the lattices A_2, D_n, $\frac{1}{2}RE_8$ and $\frac{1}{2}RE_{10}$) and with strong restrictions on $\mathbf{m}$.

III. Generalized Voronoi indexing

Fast indexing algorithms for generalized Voronoi codes based on the lattices A_2, D_n and $\frac{1}{2}RE_8$ are presented in [3]. Their common framework is summarized in Fig. 2. Contrary to [3], we have incorporated the offset $\mathbf{a}$ in the description of the inverse mapping.

Forward mapping: codevector $\mathbf{y} \rightarrow$ index $\mathbf{k}$

1. Compute $\mathbf{j} = \mathbf{y}G_\Lambda^{-1}$
2. Modify conditionally $\mathbf{j}$
3. Compute $\mathbf{k} = \mathbf{mod}(\mathbf{j}, \mathbf{m})$

Inverse mapping: index $\mathbf{k} \rightarrow$ codevector $\mathbf{y}$

1. Compute $\mathbf{u} = \mathbf{k}G_\Lambda$
2. Compute $\mathbf{z} = \mathbf{div}(\mathbf{u} - \mathbf{a}, \mathbf{m})$
3. Find the nearest neighbor $\mathbf{v}$ of $\mathbf{z}$ in Λ
4. Compute $\mathbf{y} = \mathbf{u} - \mathbf{mult}(\mathbf{m}, \mathbf{v})$

Fig. 2. Generalized Voronoi indexing in a lattice Λ.

Note that the index $\mathbf{k} = [k_1 \cdots k_n]$ is a vector of integers which verify $0 \leq k_i < m_i$ $(1 \leq i \leq n)$.

A. Condition on the existence of generalized Voronoi codes

For the inverse mapping to produce lattice points in Λ we shall verify the condition : $\mathbf{mult}(\mathbf{m}, \mathbf{v}) \in \Lambda$ where $\mathbf{m}$ is a vector of integers and $\mathbf{v}$ a lattice point. This condition implies constraints on $\mathbf{m} = [m_1, \cdots, m_n]$ which can be summarized in a matrix format as: $G_\Lambda diag(m_1, \cdots, m_n) G_\Lambda^{-1}$ is a matrix of integers.

Example: For the lattice A_2 the above condition becomes

$$G_{A_2} \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} G_{A_2}^{-1} = \begin{bmatrix} m_1 & 0 \\ \frac{m_1 - m_2}{2} & m_2 \end{bmatrix}, \qquad (5)$$

where G_{A_2} is set as in Fig. 3. We obtain that m_1 and m_2 shall have the same parity.

It can be shown that a similar condition applies for the lattices D_n, i.e. $m_1, \cdots, m_n$ shall have the same parity. And for the lattices RE_{2n} we obtain: all m_i shall have the same parity and their sum shall be a multiple of 4.

B. Conditional modification step in forward mapping

The conditional modification step in the forward mapping depends on the lattice Λ (and possibly the offset $\mathbf{a}$ [3]). It is needed, otherwise the forward mapping may not produce unique codevector labels. Note that when the modulo vector $\mathbf{m}$ has identical components, this step is not required.

$$\begin{bmatrix} 1 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \qquad \begin{bmatrix} 2 & & & \\ 1 & 1 & & \\ \vdots & & \ddots & \\ 1 & & & 1 \end{bmatrix} \qquad \begin{bmatrix} 4 & & & \\ 2 & 2 & & \\ \vdots & & \ddots & \\ 2 & & & 2 \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

(a) G_{A_2} (b) G_{D_n} (c) $G_{RE_{2n}}$

Fig. 3. Generator matrices for A_2, D_n and RE_{2n}

In what follows, we consider the case of A_2, D_n and RE_{2n}. The proposed solution generalizes the algorithms described in [3] and requires the lattices to be specified by a lower triangular generator matrix as in Fig. 3.

B.1 Conditional modification for A_2

If y belongs to the generalized Voronoi code $C(A_2, \mathbf{m}, \mathbf{a})$, we can assume that it was generated from the index $\mathbf{k} = [k_1 k_2]$ by inverse mapping. Then we can write $y_1 = u_1 - m_1 v_1$ and $y_2 = u_2 - m_2 v_2$, with $[u_1\, u_2] = [k_1\, k_2] G_{A_2}$. When applying the forward mapping to $\mathbf{y}$, we get $j_1 = k_1 - (m_1 v_1 - m_2 v_2/\sqrt{3})$ and $j_2 = k_2 - 2/\sqrt{3}\, m_2 v_2$. The role of the condition modification is therefore to ensure that $j_1 = k_1 (mod\, m_1)$ and $j_2 = k_2 (mod\, m_2)$. This suggests the following operations :

1. Compute $w = -\lfloor j_2/m_2 \rfloor$
2. Modify j_1 to $j_1 + (m_1 - m_2)w/2$

Note that to deal with integers only, this algorithm could be translated into the root coordinates of A_2 [13]. Furthermore, it can be checked easily that if $m_1 = m_2 = r$ the conditional modification is useless, because $k_i \equiv j_i (mod\, r)$.

B.2 Conditional modification for D_n and RE_{2n}

We use the same principle as for the lattice A_2, and we obtain the following sequence of operations for D_n :

1. Compute $v_i = -\lfloor j_i/m_i \rfloor$ for $2 \leq i \leq n$
2. Modify j_1 to $j_1 + \frac{1}{2}\sum_{i=2}^{n} (m_1 - m_i) v_i$

For RE_{2n} the conditional modification shall treat even and odd lattice points separately. It can be implemented as follows :

1. Compute $v_{2n} = -\lfloor j_{2n}/m_{2n} \rfloor$
2. Compute for $2 \leq i < 2n$

$$v_i = \begin{cases} -2\lfloor (j_i - \frac{m_{2n} v_{2n}}{2})/m_i \rfloor, & v_{2n} \text{ even} \\ -2\lfloor (j_i - \frac{m_{2n} v_{2n}}{2})/m_i - \frac{1}{2} \rfloor - 1, & v_{2n} \text{ odd} \end{cases}$$

3. Modify j_i to $j_i + (m_i v_i - m_{2n} v_{2n})/2$ for $2 \leq i < 2n$

4. Compute $w' = \sum_{i=2}^{2n} (m_1 - m_i) v_i$

5. Modify j_1 to $j_1 + w'/4 + (n-1) m_{2n} v_{2n}/2$ if n is odd, otherwise to

$$\begin{cases} j_1 + w'/4 + (n-1) m_{2n} v_{2n}/2, & v_{2n} \text{ even} \\ j_1 + w'/4 + (m_1 + (n-1) m_{2n} v_{2n})/2, & v_{2n} \text{ odd} \end{cases}$$

IV. Search in a Generalized Voronoi Code

We propose two alternative strategies for nearest-neighbor search in generalized Voronoi codes. One strategy essentially consists of projecting outliers on a relevant Voronoi facet in a similar fashion as in [2], while the other one exploits the lattice shell structure to sature outliers by permutation decoding as in [5], [4].

A. Projection-based saturation

A natural approach to saturate an outlier consists of reducing iteratively its radius until it falls inside the effective saturation boundary [3]. This solution is simple, yet its computational complexity is in general unbounded. To saturate outliers with a limited number of nearest-neighbor searches in the infinite lattice Λ a projection can be used.

A.1 Simplified algorithm for A_2

A fast algorithm using an explicit description of the truncation region $V_{A_2}(\mathbf{0}, \mathbf{m}) + \mathbf{a}$ in terms of lattice holes is illustrated in Fig. 4. The complete sequence of operations is shown in Fig. 5.

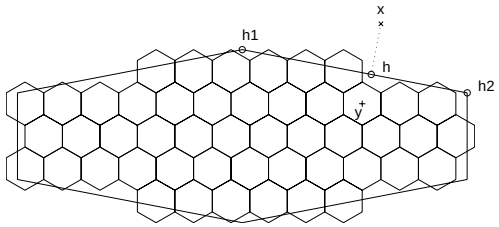


Fig. 4. Example of projection-based saturation for A_2.

An outlier $\mathbf{x}$ is projected orthogonally on the "relevant facet" of $V_{A_2}(\mathbf{0}, \mathbf{m}) + \mathbf{a}$. This facet is identified by 2 scaled and translated A_2 deep holes $\mathbf{h}_1$ and $\mathbf{h}_2$. The projection $\mathbf{h}$ is forced to lie in between these 2 points, and it may be still an outlier for $C(A_2, \mathbf{m}, \mathbf{a})$. In case $\mathbf{h}$ is an outlier its nearest-neighbor $\mathbf{z}$ in A_2 is translated adequately to get the lattice point $\mathbf{y}$, otherwise we set $\mathbf{y} = \mathbf{z}$.

The final step (translating $\mathbf{z}$ by $-\mathbf{r}$) causes the algorithm to be suboptimal, but it forces $\mathbf{z}$ to be a codevector with no additional lattice search. An alternative would be to search around $\mathbf{z}$ – for instance, by testing all 6 translations by a minimal-norm vectors, retaining those which yield a codevector, and selecting the closest to $\mathbf{x}$ – but this is more complex.

For A_2 one translation by a minimal-norm vector is enough. However this is not true in higher dimensions

Nearest neighbor search: point $\mathbf{x} \rightarrow$ codevector $\mathbf{y}$

1. Find the nearest neighbor $\mathbf{z}$ of $\mathbf{x}$ in A_2
2. If $\mathbf{z}$ is a codevector (i.e. $\mathbf{z} \in C(A_2, \mathbf{m}, \mathbf{a})$), $\mathbf{y} = \mathbf{z}$ and stop.
3. Project $\mathbf{x}$ on $\mathbf{h} \in \{V_{A_2}(\mathbf{0}, \mathbf{m}) + \mathbf{a}\}$:
   (i) Find the two closest lattices holes $\mathbf{x}_1$ and $\mathbf{x}_2$ to $\mathbf{div}(\mathbf{x} - \mathbf{a}, \mathbf{m})$ among the set of 6 points:
   $(\pm 1/2, \pm 1/2\sqrt{3})$ and $(0, \pm 1/\sqrt{3})$
   (ii) $\mathbf{h}_1 = \mathbf{mult}(\mathbf{x}_1, \mathbf{m}) + \mathbf{a}$ and $\mathbf{h}_2 = \mathbf{mult}(\mathbf{x}_2, \mathbf{m}) + \mathbf{a}$
   (iii) Compute $\alpha = \frac{(\mathbf{x} - \mathbf{h}_2)(\mathbf{h}_1 - \mathbf{h}_2)^T}{(\mathbf{h}_1 - \mathbf{h}_2)(\mathbf{h}_1 - \mathbf{h}_2)^T}$
   (iv) Saturate α such that $0 < \alpha < 1$
   (v) Compute $\mathbf{h} = \alpha \mathbf{h_1} + (\mathbf{1} - \alpha) \mathbf{h_2}$
4. Find the nearest neighbor $\mathbf{z}$ of $\mathbf{h}$ in Λ
5. If $\mathbf{z}$ is a codevector, $\mathbf{y} = \mathbf{z}$ and stop.
6. Force $\mathbf{z}$ in $C(A_2, \mathbf{m}, \mathbf{a})$:
   (i) Compute minimal-norm vector $\mathbf{r} = \mathbf{x}_1 + \mathbf{x}_2$
   (ii) Set $\mathbf{y} = \mathbf{z} - \mathbf{r}$ and stop ($\mathbf{y}$ is a codevector).

Fig. 5. Search in a generalized Voronoi code by projection (for A_2).

(e.g. for D_4) where typically the effective saturation exhibits "holes" when compared to the shaping region. In this case two successive translations may be used to really fall inside the effective overload boundary defined as the surface of $V_\Lambda(\mathbf{0}) + C(\Lambda, \mathbf{m}, \mathbf{a})$.

A.2 General case (for a lattice Λ)

The projection-based algorithm proposed for the lattice A_2 is generalized in Fig. 6. Two variants are proposed for projecting outliers:

- An orthogonal projection with a correction procedure to force projected points on the relevant Voronoi facet,
- A radial projection.

The geometry of $V_\Lambda(0)$ becomes complex as far as the dimension n increases. Therefore the use of lattice holes after an orthogonal projection is interesting only for lattices in low dimensions (typically up to $n = 4$).

The final step (a translation or a scaling) ensures that the saturation algorithm will eventually come up with a codevector. If we use a translation, we exploit the nature of generalized Voronoi codes to save a lattice decoding compared to the variant based on scaling. The value of β can be set according to $\mathbf{m}$ and d_{min}, where d_{min} is the minimal distance in Λ.

This general algorithm is suboptimal by definition. If lattice holes are not used to force the projection $\mathbf{h}$ on the relevant Voronoi facet, the algorithm requires virtually no storage.

B. Saturation by "leaders"

The projection-based saturation is limited by the fact that the shaping region $V(\mathbf{0}, \mathbf{m}) + \mathbf{a}$ and the effective overload boundary (i.e. the surface of $V(\mathbf{0}) + C(\Lambda, \mathbf{m}, \mathbf{a})$) do not coincide. A possible solution is then to specify explicitly the codevectors which define the effective overload boundary in terms of spherical shells [4], [5] or ellipsoidal shells [7], [8], [9]. The outlier saturation then involves techniques which are classical in spherical or ellipsoidal lattice quantization.

A generalized Voronoi code $C(\Lambda, \mathbf{m}, \mathbf{a})$ is then viewed as a set of (possibly incomplete) embedded spherical or ellipsoidal shells as depicted in Fig. 7. Each shell can be

Nearest-neighbor search: point $\mathbf{x} \rightarrow$ codevector $\mathbf{y}$

1. Find the nearest neighbor $\mathbf{z}$ of $\mathbf{x}$ in Λ
2. If $\mathbf{z}$ is a codevector (i.e. $\mathbf{z} \in C(\Lambda, \mathbf{m}, \mathbf{a})$), $\mathbf{y} = \mathbf{z}$ and stop.
3. Project $\mathbf{x}$ on $\mathbf{h} \in \{V_\Lambda(\mathbf{0}, \mathbf{m}) + \mathbf{a}\}$ (2 variants):
   (i) Find closest relevant vector $\mathbf{r}$ to $\mathbf{div}(\mathbf{x} - \mathbf{a}, \mathbf{m})$ and compute $\mathbf{s} = \mathbf{mult}(\mathbf{r}/2, \mathbf{m}) + \mathbf{a}$

   (ii) orthogonal projection:

   • Project $\mathbf{x}$ orthogonally on the hyperplane of $V_\Lambda(\mathbf{0}, \mathbf{m}) + \mathbf{a}$ including $\mathbf{s}$
   • Force $\mathbf{h}$ on $V_\Lambda(\mathbf{0}, \mathbf{m}) + \mathbf{a}$ (2 variants):
   – Replace $\mathbf{h}$ by the radial projection of $\mathbf{x}$ if $h \notin \{V_\Lambda(\mathbf{0}, \mathbf{m}) + \mathbf{a}\}$
   – Use an explicit description of the Voronoi facet by lattice holes

   (ii) radial projection:
   Compute $\mathbf{h} = \alpha(\mathbf{x} - \mathbf{a}) + \mathbf{a}$ with $\alpha = (\mathbf{s} - \mathbf{a})\mathbf{r}^\mathbf{T}/(\mathbf{x} - \mathbf{a})\mathbf{r}^\mathbf{T}$
4. Find the nearest neighbor $\mathbf{z}$ of $\mathbf{h}$ in Λ
5. If $\mathbf{z}$ is a codevector, $\mathbf{y} = \mathbf{z}$ and stop.
6. Otherwise (2 variants):

   scaling:
   scale $\mathbf{h}$ by a factor β: $\mathbf{h} = \beta(\mathbf{h} - \mathbf{a}) + \mathbf{a}$, and goto step 4.

   offset:
   set $\mathbf{y} = \mathbf{z} + \delta(\mathbf{z})$ where $\delta(\mathbf{z})$ is a low-energy lattice point (e.g. $\delta(\mathbf{z}) = -\mathbf{r}$ if $\mathbf{r}$ is the closest relevant vector to $\mathbf{div}(\mathbf{z} - \mathbf{a}, \mathbf{m})$), and goto step 5.

Fig. 6. Search in a generalized Voronoi code by projection (for a general lattice Λ).

partitioned into a set of (possibly incomplete) permutation codes generated by vectors referred to as "leaders" in [4], [5]. The outmost shells can be used to derive a near-optimal saturation. This requires to enumerate leaders as well as their incomplete permutations.

A saturation by leaders can be designed to achieve optimal performance if enough leaders are selected. However this advantage is mitigated by several observations:

- Leaders shall be picked among several spherical or ellipsoidal shells for good performance, because the Voronoi shaping is not exactly spherical or ellipsoidal.
- The necessary enumeration of codevectors and the selection of adequate leaders depend strongly on $\mathbf{m}$. This is problematic if we use a mixture of generalized Voronoi codes specified with different modulo vectors (as in [6]), because the saturation tables will usually differ from one code to another. This contrasts with the generality of a projection-based saturation.
- The saturation by leaders is not adapted to high-dimensional lattices (say, in dimension $n > 8$) and high rates (> 2 bits per dimension) because the number of leaders and incomplete permutations increases exponentially.

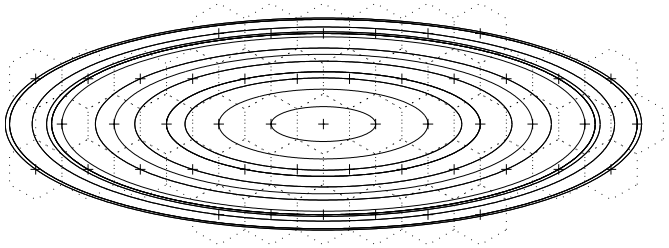


Fig. 7. Ellipsoidal interpretation of a Voronoi code (for A_2).

V. Conclusion

We presented a class of near-ellipsoidal lattice codes with fast indexing algorithms. The problem of designing efficient nearest-neighbor search algorithms was investigated and two approaches were proposed: a saturation by projection on the shaping region, and a near-spherical or near-ellipsoidal saturation by "leaders".

The focus of the paper was on algorithms. Several aspects were not considered:

- General bounds on the number of lattice searches (and possible iterations) in projection-based saturation.
- Performance evaluation for artificial memoryless and correlated Gaussian sources to estimate the shaping gain of generalized Voronoi codes over lattice codes shaped by hypercubes or hyperspheres.
- Analytical approximations of the quantization distortion.
- Further generalization of indexing algorithms to relax constraints on modulo vectors.

Note that the proposed algorithms can also be used for standard Voronoi codes. An application to wideband speech coding can be found in [6] where the lattices D_{16}, RE_{16} and $R\Lambda_{16}$ were applied with a projection-based saturation (radial projection and iterative scaling).

VI. Acknowledgment

The authors would like to acknowledge the contribution of Prof. Jean-Pierre Adoul in the early development of generalized Voronoi codes.

References

[1] J.H. Conway and N.J.A. Sloane, "A Fast Encoding Method for Lattice Codes and Quantizers," *IEEE Trans. Inform. Th.*, IT-29, no. 6, pp. 820–824, Nov. 1983.

[2] C. Pépin, *Quantification vectorielle et codage conjoint source-canal par les réseaux de points*, Ph.D. thesis, ENST, Paris, France, Dec. 1997.

[3] M. Xie, *Quantification vectorielle algébrique et codage de parole en bande élargie*, Ph.D. thesis, University of Sherbrooke, Quebec, Canada, Feb. 1996.

[4] J.-P. Adoul and M. Barth, "Nearest-neighbor algorithm for spherical codes from the Leech lattice," *IEEE Trans. Inform. Th.*, IT-34, no. 5, pp. 1188–1202, Sept. 1988.

[5] C. Lamblin and J.-P. Adoul, "Algorithme de quantification vectorielle sphérique à partir du réseau de Gosset d'ordre 8", *Ann. Télécommun.*, vol. 43, no. 3–4, pp. 172–186, 1988.

[6] S. Ragot, H. Lahdili, and R. Lefebvre, "Wideband LSF quantization by generalized Voronoi codes," submitted to Eurospeech, Aalborg, Denmark, Sept. 2001.

[7] T.R. Fischer, "Geometric source coding and vector quantization," *IEEE Trans. Inform. Th.*, vol. 35, no. 1, pp. 137–145, Jan. 1989.

[8] M.Barlaud, P. Solé, J.M. Moureaux, M. Antonini, and P. Gauthier, "Elliptical codebook for lattice vector quantization", *Proc. ICASSP*, vol. 5, pp. 590–593, Apr. 1993.

[9] J.M. Moureaux, M. Antonini, and M. Barlaud, "Counting lattice points on ellipsoids: Application to image coding," *Electon. Lett.*, vol. 31, no. 15, pp. 1224–1225, July 95.

[10] A.R. Calderbank and L.H. Ozarow, "Nonequiprobable signaling on the Gaussian channel," *IEEE Trans. Inform. Th.*, vol. 36, no. 4, pp. 726–740, July 1990.

[11] A.K. Khandani and P. Kabal, "Optimization of a lattice-based constellation for signaling over a partial response channel," *IEEE Trans. Com.*, vol. 46, no. 7, pp. 854–856, July 1998

[12] G.D. Forney, "Multidimensional constellations. II. Voronoi constellations", *IEEE Trans. on Selected Areas in Communications*, vol. 7, no. 6, pp. 941–958, Aug. 1989.

[13] J.H. Conway and N.J.A. Sloane, *Sphere Packings, Lattices and Groups, 3rd Edition*, Springer-Verlag, 1999.