Uniform distribution of points on a hyper-sphere with applications to vector bit-plane encoding

L.Lovisolo and E.A.B. da Silva

Abstract: In vector bit-plane encoding schemes, codebooks must be uniformly distributed on a hyper-sphere. Shells of regular lattices are often used, but they provide only a limited choice of number of vectors K and dimension N. The authors propose a method to generate codebooks in dimension N with arbitrary number K of vectors, almost uniformly distributed on a hyper-sphere. The uniform distribution of an arbitrary number of points on the surface of a hyper-sphere is still an open problem. Some mathematicians indeed consider it one of the mathematical challenges of the 21st century. The proposed method uses a combination of geometric and stochastic approaches to generate approximate solutions. The generated codebooks are tested in vector bit-plane encoding schemes. The results show that the proposed method is effective in generating codebooks almost uniformly distributed on a hyper-sphere.

1 Introduction

Bit-plane coding represents the state of the art in image compression schemes. For example, it is used in the recently developed JPEG-2000 standard [1]. In [2] the bit-plane concept has been generalized to vectors with the so-called successive approximation vector quantisation (SA-VQ). In SA-VQ, a vector $\boldsymbol{X}$ is approximated by a vector $\boldsymbol{X}_m$ as:

$$\boldsymbol{X}_m = \sum_{n=1}^{m} \alpha^n \boldsymbol{Y}_{i_n} \tag{1}$$

where

$$\boldsymbol{Y}_{i_n} \in C_N = \{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_K\}$$

and

$$0 < \alpha < 1$$

C_N is a codebook composed of unit-norm vectors, K is the cardinality of the codebook and m is the number of steps used in the approximation. A greedy algorithm to find the decomposition in eqn. 1 is illustrated in Fig. 1. In this figure, $\boldsymbol{X}_n$ is the vector approximation after step n, $\boldsymbol{r}_n = \boldsymbol{X} - \boldsymbol{X}_n$ is the residue after step n, and θ_n the angle between $\boldsymbol{r}_{n-1}$ and the vector $\boldsymbol{Y}_{i_n}$ chosen for the quantisation at this step—the vector $\boldsymbol{Y}_{i_n}$ from C_N with largest inner product with the residual $\boldsymbol{r}_{n-1}$. Note that $\boldsymbol{r}_0 = \boldsymbol{X}$ and $\boldsymbol{X}_1 = \alpha \boldsymbol{Y}_{i_1}$.

Such a representation is useful only if, as the number of vectors m used to approximate $\boldsymbol{X}$ is increased, the error in the approximation, $e_m = \|\boldsymbol{X} - \boldsymbol{X}_m\|$ tends to zero. It has been shown in [2] that a sufficient condition for this is

$$\begin{cases} \alpha \geq \dfrac{1}{2\cos\Theta(C_N)}, & \text{for } \Theta(C_N) \leq \dfrac{\pi}{4} \\ \alpha \geq \sin\Theta(C_N), & \text{for } \Theta(C_N) \geq \dfrac{\pi}{4} \end{cases} \tag{2}$$

IEE Proceedings online no. 20010361

DOI: 10.1049/ip-vis:20010361

Paper received 27th November 2000

The authors are with the Universidade Federal do Rio de Janeiro, PEE/COPPE/DEL/EE, Cx. P. 68504, Rio de Janeiro, RJ, 21945-970, Brazil

where $\Theta(C_N)$ is the maximum angle between any vector $\boldsymbol{X}$ in $\mathbb{R}^N$ and its nearest neighbour in C_N, that is

$$\Theta(C_N) = \max_{\|\boldsymbol{X}\|=1} \left\{ \min_{\boldsymbol{Y}_n \in C_N} [\arccos(\boldsymbol{X} \cdot \boldsymbol{Y}_n)] \right\} \tag{3}$$

In [2] it is shown that the error $e_m = \|\boldsymbol{X} - \boldsymbol{X}_m\|$ in the representation is bounded by $e_m \leq B\alpha^m$. From eqn. 2, the smaller $\Theta(C_N)$ is, the smaller is the α that grants convergence of the algorithm.

Therefore, to minimise the representation error for a given vector dimension N, one should minimise $\Theta(C_N)$. Two ways can be envisioned to obtain this result:

(i) Increase the number of vectors K that compose the codebook C_N. However, this approach would also increase the rate $R = (m \log_2 K)/N$.

(ii) Distribute the vectors of the codebook as uniformly as possible on the surface of an N-dimensional hyper-sphere. This would minimize $\Theta(C_N)$ while keeping the number of vectors K, and therefore the rate $R = (m \log_2 K)/N$ fixed. One should note that uniformity in this case is not very well defined. For a broader discussion about the meaning of uniformity, see [3]. In fact, uniformity in the sense of

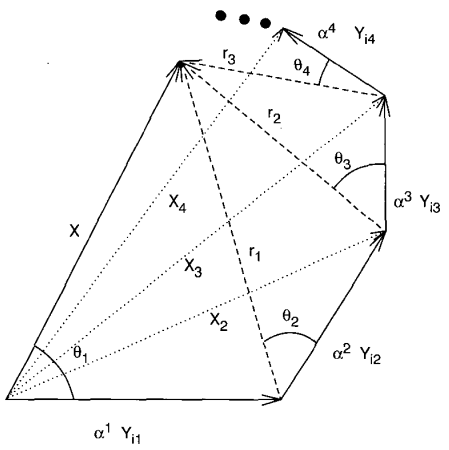

Fig. 1 *SA-VQ*

minimising $\Theta(C_N)$ is equivalent to the best covering on a hyper-sphere [4, 5].

In summary, the problem of designing 'good' codebooks for SA-VQ, and thus efficient vector bit-plane decompositions, is equivalent to distributing uniformly K points on an N-dimensional hyper-sphere. This is a problem that has been studied for centuries, but for which there is no general solution [3]. It is worth pointing out that it has applications in physics, biology, chemistry and other sciences. As mentioned in [3] it is considered one of the mathematical challenges of the 21st century.

For the purpose of distributing an arbitrary number of vectors uniformly on a hyper-sphere, three methodologies could be used:

(*a*) The algebraic [4, 5], in which orientation codebooks formed from lattices that are solutions for the packing and covering problems are used. These have been employed in [2] in the context of wavelet image coding using vector bit-planes. However, there is no solution to the general problem of distributing K points on an N-dimensional hyper-sphere. Hardin and Sloane [5] found solutions for dimensions 3, 4 and 5. The packing and covering problems are illustrated for two-dimensional space in Figs. 2*a* and 2*b* respectively, and are defined by Sloane, Hardin and Smith [5] as below.

Packing – To place n points on a sphere so as to maximise the minimal distance (or equivalently the minimal angle) between them [5].
Covering – To place n points on a sphere so as to minimise the maximum distance of any point on the sphere from the closest one of the n points [5].

(*b*) The geometric approach [3], in which the points are distributed on the surface of the sphere using geometric considerations. In [3], good asymptotic solutions for dimension 3 have been found, but there is no general method valid for higher dimensions. Two of these methods

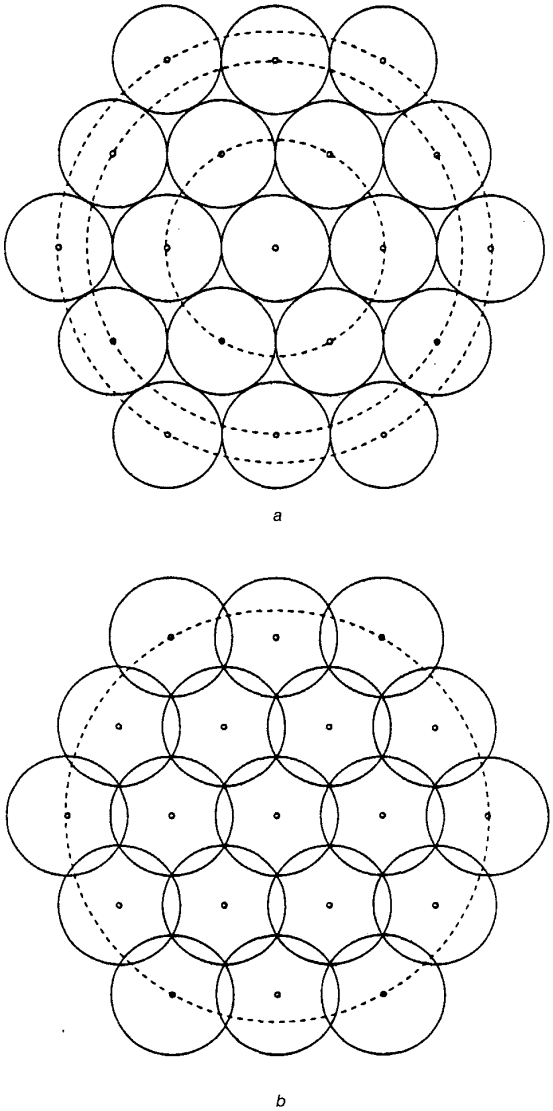

Fig. 2 *Algebraic approaches*
a Packing
b Covering

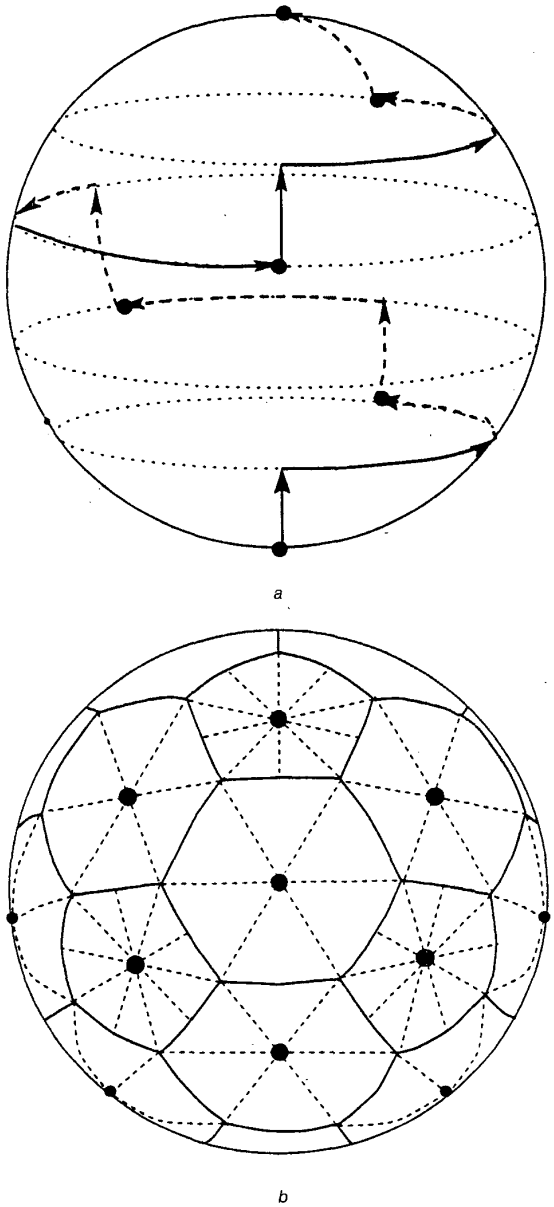

Fig. 3 *Geometric solutions*
a Construction of (six) points by a generalized spiral
b The 'ball' from 'football' or 'equilibrium of electrons': Dirichlet cells

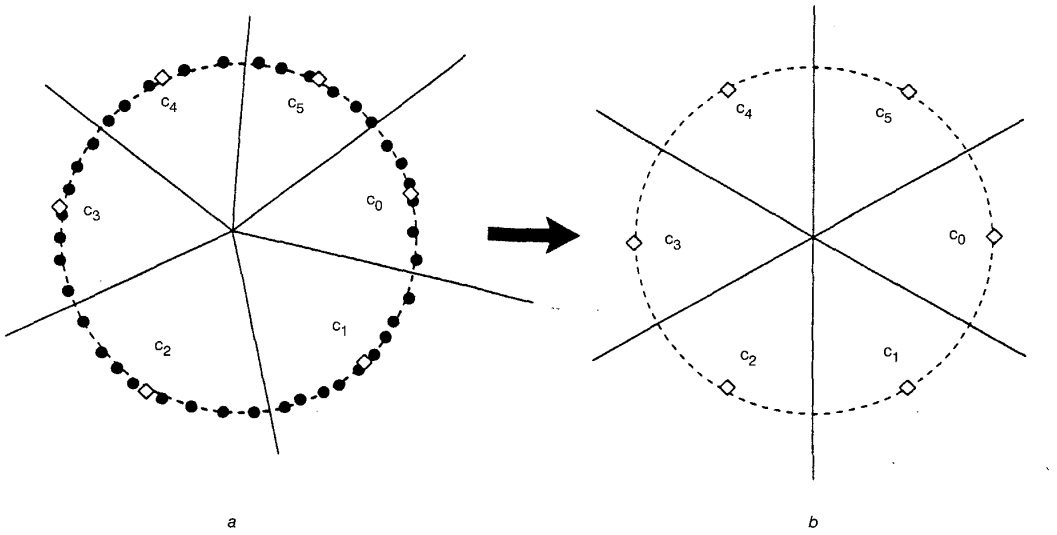

Fig. 4 *Stochastic method in two dimensions*

can be seen in Fig. 3, the generalized spiral in Fig. 3*a*, and the ball from 'football' or 'equilibrium of electrons' arrangement in Fig. 3*b*.

(*c*) A stochastic method in which an LBG-like codebook training algorithm [6] is used for vector quantisation to generate the desired codebook. The training set used is composed of vectors whose components are i.i.d. (independent identically distributed) Gaussian random variables. Since in this case their probability density function (p.d.f.) depends only on their magnitude, by normalising the vectors with a training set is obtained whose p.d.f. is uniformly distributed on the surface of a hyper-sphere. This approach is, in theory, able to generate good codebooks for any dimension N. However, the size of the training set has to be large even for moderate values of the dimension N and number of vectors K to be allocated, which makes high computational demands. This method is illustrated in Fig. 4 for a two-dimensional case. Fig. 4*a* shows an initial codebook which is non-uniformly distributed (squares) and a uniformly distributed (random) training set (circles). After a number of iterations of the LBG algorithm, the final codebook is uniformly distributed, as illustrated in Fig. 4*b*.

In this paper, a method is proposed that borrows characteristics from geometric and stochastic methods. The initial codebook and training set is designed using a novel geometric method, valid for any arbitrary dimension N and number of vectors K. The LBG algorithm is then employed to improve the initial codebook.

2 Distributing points on a hyper-sphere

First, the proposed geometric procedure (GM) is introduced, mostly based on approximations that are asymptotically valid. The GM is employed to generate initial codebooks (IC) and training sets (TS) that are used in an LBG-like algorithm.

2.1 Proposed method

Consider first the representation in spherical coordinates of a point p in $\mathbb{R}^N$ belonging to the surface of an N-dimensional hyper-sphere of unit radius. The point is given by

$$p = \{1, \omega_1, \omega_2, \ldots, \omega_{N-1}\}$$
$$\text{where } 0 \leq \omega_i \leq \pi \qquad \text{for } i \in \{1, \ldots, N-2\}$$
$$\text{and } 0 \leq \omega_{N-1} < 2\pi \tag{4}$$

From the spherical coordinates, Cartesian coordinates can easily be computed as

$$\begin{aligned}
x_1 &= \cos\omega_1 \\
x_2 &= \sin\omega_1 \cos\omega_2 \\
x_3 &= \sin\omega_1 \sin\omega_2 \cos\omega_3 \\
&\vdots \\
x_{N-2} &= \sin\omega_1 \sin\omega_2 \ldots \sin\omega_{N-3} \cos\omega_{N-2} \\
x_{N-1} &= \sin\omega_1 \sin\omega_2 \ldots \sin\omega_{N-2} \cos\omega_{N-1} \\
x_N &= \sin\omega_1 \sin\omega_2 \ldots \sin\omega_{N-2} \sin\omega_{N-1}
\end{aligned} \tag{5}$$

Fig. 5 illustrates this for a three-dimensional space.

The main supposition in the proposed method is the following: when the number of vectors K is large, there is one set of vectors uniformly distributed on an N-dimensional hyper-sphere that defines a tiling of the hyper-sphere

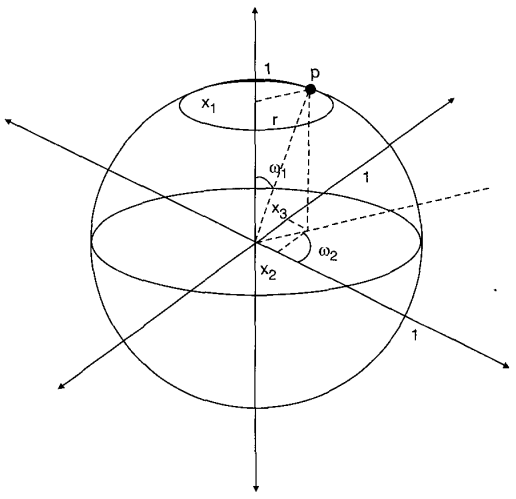

Fig. 5 *A point on a hyper-sphere in the space* $\mathbb{R}^3$

by identical hypercubes of dimension $N-1$. Considering that each hypercube edge has length δ in Euclidean space, then it can be defined by small variations $\Delta\omega_1, \Delta\omega_2, \ldots, \Delta\omega_{N-1}$, of the spherical coordinates, that is

$$\delta = \Delta\omega_1 \quad \text{and} \quad \delta = \Delta\omega_j \prod_{i=1}^{j-1} \sin\omega_i, \quad \text{for } j = 2, \ldots, N-1 \quad (6)$$

Thus given δ and a point $\omega_1, \ldots, \omega_{N-1}$, the values of $\Delta\omega_1, \Delta\omega_2, \ldots, \Delta\omega_{N-1}$ follow from eqn. 6. This suggests the following algorithm for allocating the points:

compute $\Delta\omega_1$ from eqn. 6
for $\omega_1 = \Delta\omega_1/2$ **to** π in **increments** of $\Delta\omega_1$ **do**
  compute $\Delta\omega_2$ from eqn. 6
  for $\omega_2 = \Delta\omega_2/2$ **to** π in **increments** of $\Delta\omega_2$ **do**
    ⋱
    compute $\Delta\omega_{N-2}$ from eqn. 6
    for $\omega_{N-2} = \Delta\omega_{N-2}/2$ **to** π in **increments** of $\Delta\omega_{N-2}$ **do**
      compute $\Delta\omega_{N-1}$ from eqn. 6
      for $\omega_{N-1} = \Delta\omega_{N-1}/2$ **to** 2π in **increments** of $\Delta\omega_{N-1}$ **do**
        for $i = 1 \ldots N-1$ **do**
          compute $x_i = \sin\omega_1 \ldots \sin\omega_{i-1} \cos\omega_i$
        end
        compute $x_N = \sin\omega_1 \ldots \sin\omega_{N-2} \sin\omega_{N-1}$
      end
    end
  end
end

The value of δ is computed by considering that if the K hypercubes, each one of area δ^{N-1}, provide a tiling of the hyper-sphere, then the sum of the hypercubes areas is equal to the area of the hyper-sphere (A_N), that is:

$$A_N = K\delta^{N-1} \quad (7)$$

The area A_N of an N-dimensional hyper-sphere of unit radius can be computed by [4]:

$$A_N = \frac{N\pi^{N/2}}{(N/2)!}, \quad \text{for } N \text{ even} \quad (8)$$

$$A_N = \frac{N2^N\pi^{(N-1)/2}\left[\frac{(N-1)}{2}!\right]}{N!}, \quad \text{for } N \text{ odd} \quad (9)$$

From these equations we can compute the value of δ based on the area of an N-dimensional hyper-sphere of unit radius, A_N, and the number of vectors, K, that we want to distribute.

Since both eqn. 6 and the tiling are only asymptotically valid (large K), the value of δ provided by the tiling may give rise to a number of points $K' \neq K$. This can be corrected by making $\delta = \delta'(K'/K)^{1/N-1}$ and applying the procedure iteratively until $K' = K$.

2.2 Evaluating distributions of points

The method for allocating the points, proposed in subsection 2.1 was evaluated based on the angles between the vectors of the allocated codebook. Specifically three angles were studied that will be addressed as:

- *maximum*: the maximum angle between any two nearest vectors of the codebook;
- *minimum*: the minimum angle between any two nearest vectors of the codebook;
- *average*: the average of the angles between any two nearest vectors of the codebook.

Fig. 6 shows the angles between the vectors allocated by the proposed method for different cardinalities of a dictionary in three dimensions. It can be seen that the average angle approaches the maximum asymptotically. Fig. 7 depicts the points allocated by the proposed method for a cardinality $K = 90$ in three-dimensional space ($N = 3$). It is termed a codebook (3,90). Fig. 8 shows the points gener-

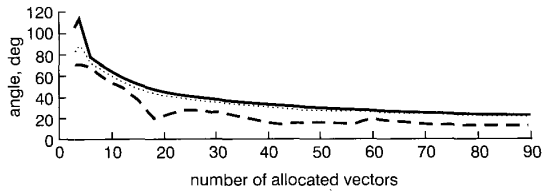

Fig. 6 *Maximum, average and minimum angle between the vectors allocated by the proposed method (GM) for $N = 3$ and different cardinalities*

—— maximum
···· average
– – – minimum

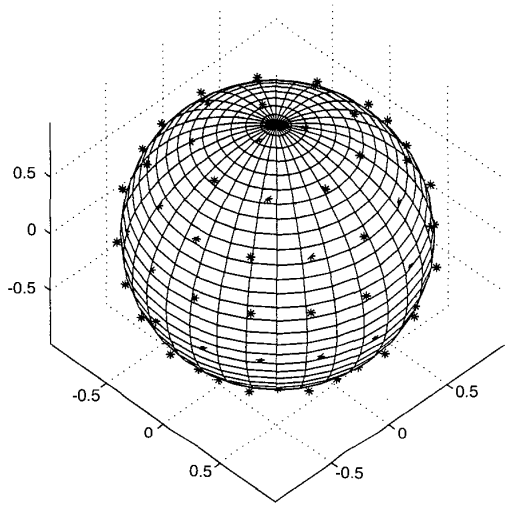

Fig. 7 *Points generated by the method for a codebook (3,90)*

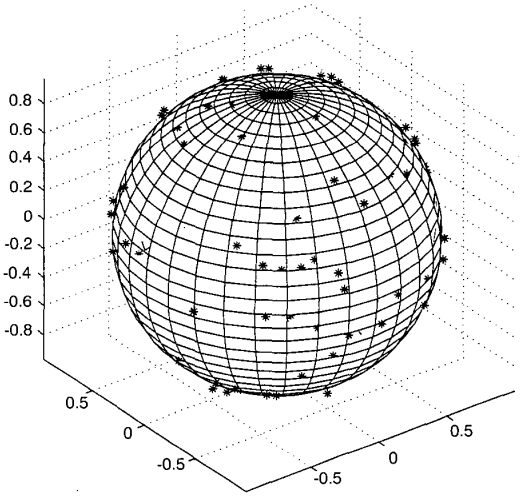

Fig. 8 *Points generated by i.i.d. Gaussian random variables for a codebook (3,90)*

ated by i.i.d. Gaussian random variables for the same dimension N and number of points K. We can see clearly that the proposed geometric method generates points much more uniformly distributed than the one based on i.i.d. Gaussian random variables.

2.3 Improvements

Since the proposed method generates asymptotically valid approximations (large K), it should be good enough to generate a training set (TS) for the LBG algorithm, given the fact that training sets consist of a large number of vectors. This circumvents the problem of the very large TS required by the stochastic approach. This is because, using the proposed geometric method, the same degree of uniformity can be achieved using a number of vectors orders of magnitude smaller than the one normally employed in the stochastic approach, as suggested by Figs. 7 and 8. It has also been verified that it generates good initial codebooks (IC). The motivation to verify this last possibility came from the fact that the standard LBG algorithm converges to a local minimum (close to the initial codebook) that may not be the optimal one. Given the fact that the generated distributions using GM are already almost uniformly distributed, a local minimum near to them has a better chance of being optimal than a randomly chosen solution for the problem under study.

In these experiments, the distortion metric used was the mean squared error. In addition, a few adjustments had to be made to the LBG algorithm. Since the centroid of a region on a hyper-sphere is not on it, in the LBG algorithm the magnitude of the centroids had to be scaled in every iteration to keep it on the surface of the hyper-sphere. One should note that due to this scaling procedure the LBG centroid optimality condition is violated; therefore, convergence is not strictly guaranteed. It is therefore supposed that convergence is achieved in the LBG iteration in which total distortion reaches a minimum.

To assess the effectiveness of GM in generating both the TS and the IC, the performance of codebooks generated by the LBG algorithm for four combinations of TS and IC have been evaluated. They are referenced as in Table 1. Column IC contains the method used to generate the initial codebook, and column TS the method used to generate the training set. In the table GM stands for the geometric method and IIDG for i.i.d. Gaussian random variables as explained earlier.

Their performances were compared in terms of number of iterations needed for the LBG algorithm to converge, as well as maximum, average and minimum angles between two nearest vectors in the codebook. Two observations were made:

(i) The MM and IM types generate more uniform codebooks, with the closest maximum and minimum angles. For $N=3$ and $K=12$ ((3,12) codebooks) and training sets composed of 9600 vectors, the maximum, average and minimum angles are shown in Table 2.

(ii) the MM type simulation converges faster to the final codebook. This can be observed by the number of required iterations for the LBG to converge. Some of those numbers are shown in Table 3 for several dictionaries in three dimensions using training sets 800 times larger than the codebooks in all cases. On average, compared to the MM type, the II type needs 114% more iterations, the IM 20% more, and the MI 68% more.

Table 1: Methods for generating codebooks

Method name	IC	TS
MM	GM	GM
IM	IIDG	GM
MI	GM	IIDG
II	IIDG	IIDG

Table 2: Maximum, average and minimum angles for a codebook (3,12) generated by the four different methods with a fixed size TS of 9600 vectors

Method name	Maximum angle	Average angle	Minimum angle
MM	63.28°	63.07°	62.64°
IM	63.29°	63.01°	62.82°
MI	61.79°	60.33°	58.09°
II	62.07°	60.36°	58.16°

Table 3: Number of iterations needed by the LBG algorithm to converge for the four methods to generate some codebooks in three dimensions and TS 800 times larger than the cardinality of the codebook

		Number of iterations by method			
Codebook size	Training set size	MM	MI	IM	II
4	3200	8	41	7	30
6	4800	18	23	14	47
8	6400	35	28	31	126
12	9600	33	97	32	62
20	16000	42	63	86	95
24	19200	61	105	73	122
30	24000	80	198	93	271
35	28000	63	69	86	96
40	32000	93	68	61	109
45	36000	74	132	101	92
50	40000	144	187	157	236
60	48000	80	140	88	261
70	56000	122	124	102	94
80	64000	95	126	153	187
90	72000	136	127	123	213
96	76800	87	104	131	133

Taking the above two points into consideration, it can be concluded that the MM type is clearly the best choice, that is, the geometric method was effective in generating the TS and the IC. From the experimental results obtained it can be seen that the choice of the TS is more critical to the quality of the final codebook, while the IC influences the speed of convergence. It has also been observed that as the size of the TS increases, the performance of the generated codebooks increases asymptotically. For example, for (4,24) codebooks, a TS 5000 times larger than the IC generates a codebook with performance measured by the maximum, minimum and average angles, very close to the one in [5].

3 Performance of the generated codebooks

The generated codebooks were evaluated using the SA-VQ algorithm described in Section 1. The input data was composed of vectors with a uniform statistical distribution on the surface of a hyper-sphere. Each codebook was evaluated by the average distortion after m steps (see eqn. 1). Codebooks were generated in dimensions 3, 4 and 5 and were compared to the codebooks from [5]. Table 4 lists the results for codebooks that are the best known solutions for the covering and packing problems [5] for various K and $N \in \{3, 4, 5\}$. Also listed are the results for the codebooks generated by the MM method for the same pair (N,K). The input data was quantised using SA-VQ for each pair (N,K). The mean squared error (mse) for unit variance input shown was computed for the value of α (see eqn. 1) for which it is a minimum. These data show that the proposed method consistently generates codebooks with good performance, that is, low values of mse.

Fig. 9 shows a comparision of the mse for the codebook (4,24) from [5] ($D_{4_{S_1}}$, first shell of the lattice D_4) with the mse of the codebook (4,24) generated by the method MM, for α in the range [0.55, 0.7], using an eight-step SA-VQ algorithm. An equivalent performance is observed for the generated codebook and the one from [5] for all values of α.

The performance difference in terms of mse between the two previous codebooks analysed in Fig. 9 can be observed in more detail in Fig. 10. Fig. 10 shows the performance difference for $\alpha \in [0.50, 0.90]$ and number of steps m ranging from 2 to 8. Two points should be noted:

(i) the difference is very small;
(ii) in the region in which both codebooks have the best performance, $\alpha \in [0.50, 0.60]$, the difference vanishes.

From these results it is argued that the generated codebooks perform consistently well for a large range of values of α and m.

The generated codebooks were also tested in the framework of the successive approximation wavelet vector quantisation (SA-W-VQ) algorithm [2, 7] for the image 'Lena' at 0.5 bits/pixel for different values of α. The SA-W-VQ algorithm codes the wavelet transform of an image using vectors bit-planes, as well as the concept of zero-trees and arithmetic coding.

Table 5 shows the PSNR for the codebook formed by the first shell of the regular lattice D_4, best known 'packing' and 'covering' in dimension four ($D_{4_{S_1}}$) with 24 points, and a codebook generated with the same pair $(N,K) = (4,24)$, by the proposed method with MM type simulation, designated MM(4,24). Fig. 11 shows the PSNR for the same codebooks $D_{4_{S_1}}$ and MM(4,24) against α, for all its allowed range. Fig. 12 shows the performance of the $E_{8_{S_1}}$ codebook formed by the first shell of lattice E_8, compared with a codebook with the same (N,K), the MM(8,240). Some of its points are also listed in Table 5.

Table 4: Mean squared error for different codebooks, for unit variance input

(N,K)	Codebook generation	SA-VQ steps	α optimum	mse
(3,12)	packing	9	0.59	0.000052
(3,12)	MM method	9	0.58	0.000039
(3,12)	packing	6	0.59	0.001331
(3,12)	covering	6	0.58	0.001094
(3,12)	MM method	6	0.58	0.001094
(3,15)	packing	6	0.57	0.000954
(3,15)	covering	6	0.56	0.000797
(3,15)	MM method	6	0.56	0.000797
(3,21)	packing	6	0.55	0.000561
(3,21)	covering	6	0.55	0.000562
(3,21)	MM method	6	0.55	0.000564
(4,24)	packing	6	0.59	0.001490
(4,24)	MM method	6	0.60	0.001625
(4,50)	packing	6	0.56	0.000671
(4,50)	MM method	6	0.50	0.000667
(5,64)	packing	6	0.59	0.001326
(5,64)	MM method	6	0.59	0.001311
(5,100)	packing	6	0.57	0.000867
(5,100)	MM method	6	0.57	0.000833

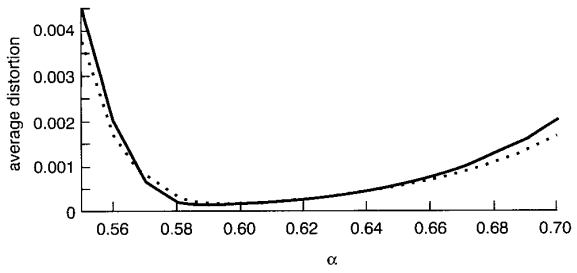

Fig. 9 *mse for SA-VQ with 8 steps for codebook (4,24)*
—— D_4
···· generated

Table 5: PSNR (dB) for the SA-W-VQ for some values of α, for $D_{4_{S_1}}$, MM(4,24), $E_{8_{S_1}}$ and MM(8,240), for the image 'Lena' at 0.5 bpp

Codebook	α 0.55	0.62	0.71
$D_{4_{S_1}}$	32.06	31.54	31.20
MM(4,24)	32.09	31.57	31.20
$E_{8_{S_1}}$	31.46	31.72	31.22
MM(8,240)	31.02	31.62	31.16

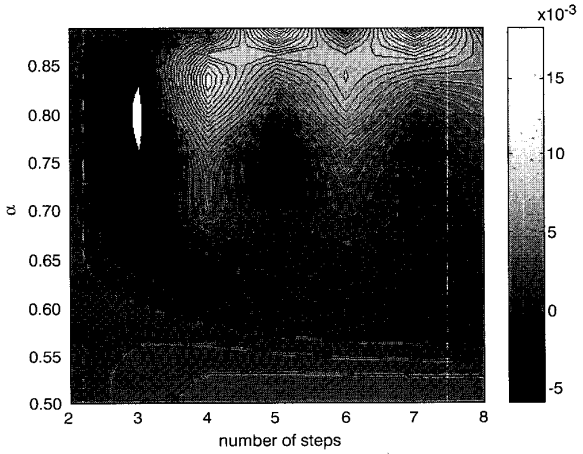

Fig. 10 *mse difference for the codebook (4,24) from [5] and the one generated by the method MM*

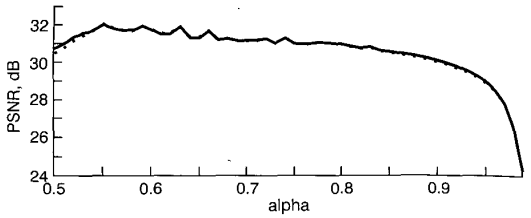

Fig. 11 *PSNR (dB) of the image 'Lena' at 0.5 bit/pixel for the $D_{4_{5_1}}$ and a codebook with the same (N,K), MM(4,24)*

—— $D_{4_{5_1}}$
···· MM

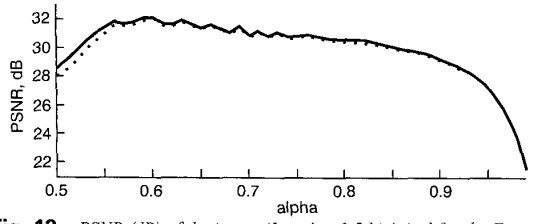

Fig. 12 *PSNR (dB) of the image 'Lena' at 0.5 bit/pixel for the $E_{8_{5_1}}$ and a codebook with same (N,K), MM(8,240)*

—— $E_{8_{5_1}}$
···· MM

In these cases roughly the same PSNR performance (within a few hundredths of a dB) was obtained, and this is also an indication of the effectiveness of the proposed method. From the results shown it can be seen that the proposed method effectively generates codebooks with arbitrary dimension N and cardinality K for vector bit-plane encoding applications.

4 Conclusions

A method of allocating an arbitrary number of points almost uniformly on the surface of a hyper-sphere has been proposed. This method was used to generate training sets and initial codebooks to be employed in an LBG algorithm. The computational demands of the LBG algorithm were greatly reduced in comparison to those of the stochastic approach. Therefore the proposed method effectively distributes an arbitrary number K of vectors almost uniformly on an N-dimensional hyper-sphere.

In those cases where there are known solutions to the packing and covering problem, it was seen that the proposed method generates codebooks as good as those. Therefore, the proposed method is suited to generating codebooks to be used in any vector bit-plane coding scheme of arbitrary dimension and cardinality.

5 References

1 CHRISTOPOULOS, C. (Ed.): 'JPEG2000 Verification Model 5.3 (Technical Description)', JPEG2000 Web site, JPEG2000 core and VM reflectors
2 CRAIZER, M., DA SILVA, E.A.B., and RAMOS, E.G.: 'Convergent algorithms for successive approximation vector quantization and applications to wavelet image compression', *IEE Proc., Vis. Image Signal Process.*, June 1999, **146**, (3), pp. 159–164
3 SAFF, E.B., and KUIJLAARS, A.B.J.: 'Distributing many points on a sphere', *Math. Intelligencer*, 1997, **19**, (1), pp. 5–11
4 CONWAY, J., and SLOANE, N.: 'Sphere packings, lattices and groups', *in* 'A series of comprehensive studies in mathematics' (Springer Verlag, New York, 1993, 2nd edn.)
5 NEIL, J.A., SLOANE, R.H., and HARDIN, W.D.S.: 'Neil J. A. Sloane: Home Page', http://www.research.att.com/~njas/, February 2000, and subpages on Covering and Packing
6 GERSHO, A., and GRAY, R.M.: 'Vector quantization and signal compression' (Kluwer Academic Publishers, Boston/Dordrecht/London, 1992, 1st edn.)
7 DA SILVA, E.A.B., SAMPSON, D.G., and GHANBARI, M.: 'A successive approximation vector quantizer for wavelet transform image coding', *IEEE Trans. Image Process.*, February 1996, **5**, (2), pp. 299–310