RESEARCH ARTICLE OPEN ACCESS

"Investigative Compression Of Lossy Images By Enactment Of Lattice Vector Quantization Technique With D4 Lattice System & Spherical Truncation

Anshuma Patel*, Pallavi Choudhary**
*(Department of Electronics & Communication Engineering, SBITM, RGPV, Betul-460001)
** (Department of Electronics & Communication Engineering, SBITM, RGPV, Betul-460001)

ABSTRACT
In the digital era we live in, efficient representation of data generated by a discrete source and its reliable transmission are unquestionable need. In this work we have focused on source coding taking image as source. Lattice Vector Quantization (LVQ) can be used for source coding as well as for channel coding. (LVQ) with Generator Matrix (GM) and codebook is implemented. When implementation using codebook is done, two codebooks are constructed, one with 256 lattice points that are closest to (0,0,0,0) and another with 256 lattice points that are closest to (1,0,0,0). Energy for both the codes is calculated. When we compare the energy of both the codes we find that codes centered at a non lattice point is lower energy code.
Keywords – Lossy Image Compression, VQ, LVQ, GM.

I. INTRODUCTION

The process by which data representation is accomplished is called source encoding. The device that performs the representation is called a source encoder. While for reliable transmission channel encoder and channel decoder are required. Source coding and channel coding both are very important process of any digital communication systems. Lattice Vector Quantization (LVQ) can be used for source coding as well as for channel coding. In this paper we have focused on source coding taking image as source.

Image compression is a process of obtaining a compact representation for the image by reducing the number of bits used to represent an image sample without any reduction in the perceptual quality. Image coding can be lossy or lossless. Run length encoding and entropy coding are the methods for lossless image compression. Transform coding, where transform such as Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT) applied, followed by quantization and entropy coding can be cited as a method for lossy image compression. Due to the increasing traffic caused by multimedia information and digitized form of representation of images; image compression has become a necessity. Uncompressed multi -media (graphics, audio, Video) data requires considerable storage capacity and transmission bandwidth despite rapid progress in mass storage density. The recent growth of data intensive multimedia-based web applications have not only the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology.

II. LVQ: A BRIEF REVIEW

In lattice-based vector quantization, from the finite subset of regular lattice points the codebook is constructed [1]. The symmetry of regular lattice makes the encoding and decoding very simple, as selection and indexing of the nearest codeword for a given input point becomes very straightforward. Furthermore, the code words are not necessarily stored; they can be generated by a set of simple algebraic rules. Thus, lattice based quantization offers an enormous reduction in the encoding complication and eliminates the storage requirement; these are the two inherent problems in implementing a vector quantizer.

The quantization scheme requires knowledge of the codebook by both the encoder and the decoder prior to the process of quantization. In most cases, particularly in VQ to satisfy this requirement the codebook is also sent through the communication channel, or included in the compressed file. As a result, the overall compression ratio may be significantly reduced, especially for large codebooks or high dimensionality of vectors in the case of VQ. Lattice quantization is free from this drawback because it does not require transmission of the complete codebook due to its regular structure and as a result, it is possible to generate the codebook independently at both the encoder and decoder ends. Usually in a lattice based vector quantizer, the lattice is truncated such that the desired number of lattice points fall inside the boundary. Probability density function (pdf) of given source, only a few of these lattice points are used. Efficient algorithms exist for implementing a lattice quantizer. In the existing methods, indexing requires excessive storage or

complex enumeration algorithms. Our focus is on the efficient indexing method.

II. METHODOLOGY

The lattice based vector quantizer encodes the source vectors by mapping them into the lattice points. Since lattice is a regular structure there is no need to generate and store the codebook. It makes the encoding and decoding very simple as selection of the nearest codeword for a given input becomes very straightforward. Following steps are involved in the designing of lattice vector quantizer-

2.1 Optimum Lattice

The first question to be addressed in LVQ is the choice of the optimum lattice. This choice addresses the problem of best space covering by overlapping spheres. This problem finds a strong analogy with the sphere packing theory, where it is searched to arrange the maximal number of equal non overlapping spheres in a given volume in n-dimensional space. The best packing lattice will be the one providing the densest packing of identical spheres together [2]. The best packing lattices in different dimension are as follows

Table 2.1: Densest packing lattice in various dimensions

Dimension	Densest Packing Lattice
1	$Z = \Lambda_1$
2	$A_2 = \Lambda_2$
3	$A_3 = D_3 = \Lambda_3$
4	$D_4 = \Lambda_4$
5	$D_5 = \Lambda_5$
6	$E_6 = \Lambda_6$
7	$E_7 = \Lambda_7$
8	$E_8 = \Lambda_8$
9	Λ_9
10	Λ_{10} (P_{10c})
12	K12
16	$BW_{16} \approx \Lambda_{16}$
24	Leech$\approx \Lambda_{24}$

2.2 Truncation of Lattice

In order to obtain best trade off distortion rate, we must scale and truncate the lattice suitably. To do this, we need to know how many lattice points lie within the truncated area. Hence we need to know the shape of the truncated area. Generally, the truncation or lattice support is defined by means of a norm N(x) of the lattice points which should be less than a given value K:

$$\Lambda_k = \{x = (x_1, x_2, \cdots, x_n) \in \Lambda | N(x) \le K\} \cdots\cdots (1)$$

The truncation shape is spherical if N is the Euclidean norm or l_2 norm, or pyramidal if the N is l_1, or rectangular if N is the maximum norm i.e. the maximum absolute value of the lattice vector components. Also other, more general norms can be considered. The l_1 and l_2 norms defined respectively by

$$l_1 = \sum_{i=1}^{n} |x_i|$$

$$l_2 = \sum_{i=1}^{n} \left| x_i^2 \right| \cdots (2)$$

In truncating lattice points, two parameters should be defined viz (i) the shape of the boundary and (ii) the radial parameter. For minimum distortion, the shape of the boundary is the shape of the contour of constant probability density function (PDF). The shape of the truncated area or codebook is determined according to the statistics of the source. Most sub band image data can describe by a generalized Gaussian PDF or Laplacian PDF. When the signal to be compressed has Gaussian distribution, the surfaces of equal probability are ordinary spheres. The truncated area is then spherical. When the source signal has Laplacian distribution, the contour is pyramidal. Mathematical and graphical representations of Gaussian and Laplacian distribution are as follows:-

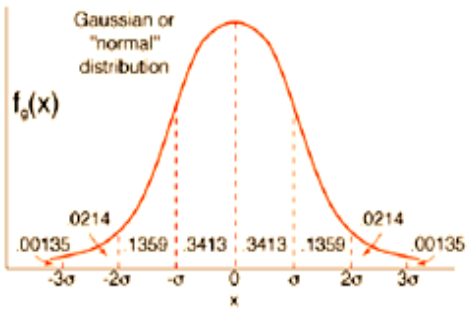
Gaussian distribution:

$$f_g(x) \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-a)^2} / 2\sigma^2 \cdots\cdots (3)$$

With mean a and σ standard deviation.

Pyramidal distribution

$$f_p(x) = \frac{\lambda}{2} e^{-\lambda|x|} \cdots\cdots (4)$$



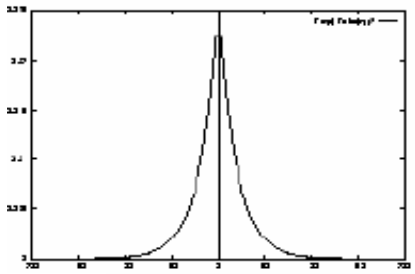


Fig2.1: Gaussian distribution and Pyramidal distribution

The isonorms defined by the norm have, in the case of two dimensions, a circular shape as shown in fig 2.1. The isonorms have the shape of spheres for a three- dimensional space and of hyper spheres for dimensions higher than three. Similarly, the isonorms defined by the norm form pyramids or hyperpyramids [3]. It is then necessary to determine the number of points lying on a hypersphere or on a hyper pyramid.

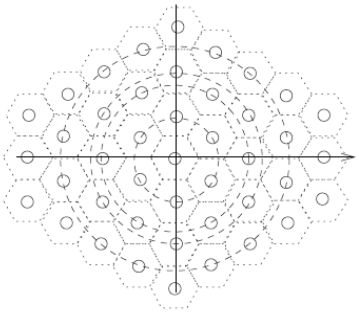


Fig 2.2: Representation of the circular truncation for hexagonal lattice (A2)

Fig 2.3 shows pyramidal truncation with code radius (m) =2 and Z2 cubic lattice number of lattice points on pyramidal truncation will be 13 i.e. size of the codebook 13 [4]. From figure 3 codebook C = {Y1, Y2, ----------, Y13}, where Y1, ---------- ------ Y13 are code vectors. Lattice code vectors on different shells are Cs (0) = 1, Cs (1) = 4 and Cs (2) = 8. Counting of lattice points is explained in detail in the next topic.

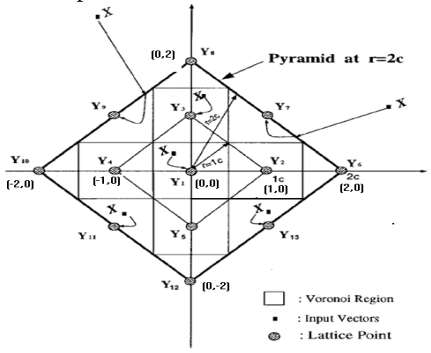


Fig2.3: Representation of pyramidal truncation for two dimensional cubic lattice (Z2)

The truncation shape is rectangular if the norm is maximum norm i.e. the maximum absolute value of the lattice vector components [5].

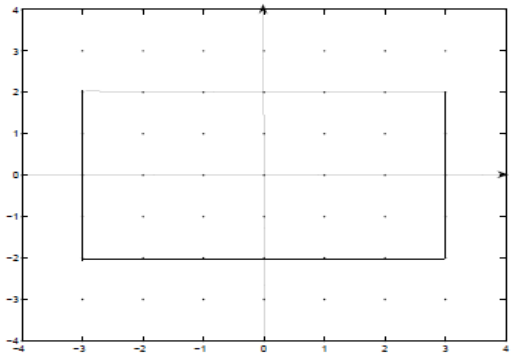


Fig 2.4: Representation of the rectangular truncation for two dimensional cubic lattices (Z2)

A given norm defines, in addition to the lattice truncation, the lattice shell, as the set of lattice points that have the same norm value, K:

$$\Lambda_k = \left\{ x = (x_1, x_2, \cdots, x_n) \in \Lambda \middle| N(x) = K \right\} \cdots\cdots (3)$$

Consequently, the lattice truncation can be seen as a union of lattice shells. The truncated lattice points must be scaled to achieve minimum distortion. The best scaling is found by repeated experiments.

2.3 Counting lattice points

Successful application of a lattice is only possible if there exist enough lattice points available for quantization. The use of lattice truncations as quantizers implies knowing the number of lattice code vectors inside the considered truncation. Following the definition of a lattice truncation as a union of shells, counting the lattice points reduces to finding expressions for the cardinality of a shell, i.e. the number of lattice points at a given distance from the origin, under the specified norm. The solution of this problem is given by the theta functions for l_2 norm for many standard lattices in [6]. In [7] the theta functions have been generalized for the norm lp and in [8], [9] for weighted l_2 norms. In [10] the theta series approach is used to count the lattice points on spherical (l_2 norm) shells and generalized to pyramidal (l_1 norm) shells. Theta function of the lattice Λ is defined as

$$\Theta_\Lambda(z) = \sum_{m=0}^{\infty} N_m q^m \cdots\cdots\cdots\cdots (5)$$

Where N_m is the number of lattice vectors with norm squared m (i.e. the number of lattice points at a distance m from the origin), z is a real number and $q=e^{\pi i z}$

Theta functions of some lattices can be specified in terms of the classical Jacobi theta functions θ_2, θ_3 and θ_4, which are defined as follows:

$$\theta_2(z) = 2\sum_{m=0}^{m=\infty} q^{(m+1(1/2))^2}$$
$$= 2q^{1/4} + 2q^{9/4} + 2q^{25/4} + \cdots$$
$$= 2q^{1/4}(1 + q^2 + q^6 + q^{12} + q^{20} + \cdots) \cdots\cdots\cdots\cdots (6)$$

$$\theta_3(z) = 1 + 2\sum_{m=1}^{\infty} q^{m^2}$$
$$= 1 + 2q + 2q^4 + 2q^9 \cdots$$
$$= 1 + 2q(1 + q^3 + q^8 + q^{15} + \cdots) \cdots\cdots\cdots\cdots (7)$$

$$\theta_4(z) = 1 + 2\sum_{m=1}^{\infty} (-q)^{m^2}$$
$$= 1 - 2q + 2q^4 - 2q^9 + \cdots$$
$$= 1 - 2q(1 - q^3 + q^8 - q^{15} + \cdots). \cdots\cdots\cdots\cdots (8)$$

Theta function of some lattices-

$$\Theta_{A_2}(z) = \theta_3(z)\theta_3(3z) + \theta_2(z)\theta_2(3z) \cdots\cdots\cdots\cdots (9)$$

$$\Theta_{Z^n}(z) = (\theta_3(z)^n) \cdots\cdots\cdots\cdots (10)$$

$$\Theta_{D_4}(z) = \frac{1}{2}(\theta_3(\frac{z}{2})^4 + \theta_4(\frac{z}{2})^4) \cdots\cdots\cdots\cdots (11)$$

$$\Theta_{D_n}(z) = \frac{1}{2}(\theta_3(\frac{z}{2})^n + \theta_4(\frac{z}{2})^n) \cdots\cdots\cdots\cdots (12)$$

$$\Theta_{E_8}(z) = \frac{1}{2}(\theta_2(\frac{z}{2})^8 + \theta_3(\frac{z}{2})^8 + \theta_4(\frac{z}{2})^8) \cdots\cdots\cdots\cdots (13)$$

The first 50 coefficients of the Theta series for A_2 are shown in Table 2.2 and Figure 2.5

Table2.2: The first 50 coefficients of the Theta series with starting point at zero for the A_2 lattice

1, 6, 0, 6, 6, 0, 0, 12, 0, 6, 0, 0, 6, 12, 0, 0, 6, 0, 0, 12, 0, 0, 0, 6, 0, 6, 12, 0, 0, 12, 0, 0, 0, 6, 12, 0, 12, 0, 0, 0, 12, 0, 0, 0, 0, 6, 18

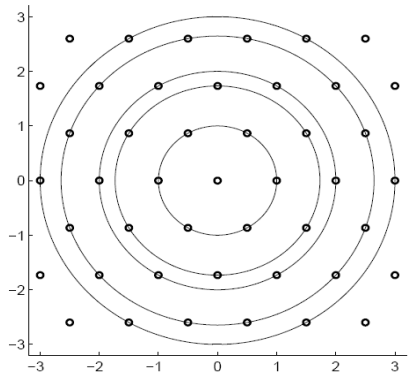


Figure 2.5: The first 6 non-zero shells of A_2 is shown as large circles (incl. the one at the origin). Notice that the number of points lying on each circle agrees with the corresponding coefficient of the Theta series. [13]

The first 50 coefficients of the Theta series for D4 are shown in Table 3

Table 2.3: The first 50 coefficients of the Theta series with starting point at zero for the D4 lattice

1, 0, 24, 0, 24, 0, 96, 0, 24, 0, 144, 0, 96, 0, 192, 0, 24, 0, 312, 0, 144, 0, 288, 0, 96, 0, 336, 0, 192, 0, 576, 0, 24, 0, 432, 0, 312, 0, 480, 0, 144, 0, 768, 0, 288, 0, 576, 0, 96, 0

The first 10 coefficients of the Theta series for E8 are shown in Table 4. [13]

Table 2.4: The first 10 coefficients of the Theta series for the E8 lattice

240, 2160, 6720, 17520, 30240, 60480, 82560, 140400, 181160, 272160

2.4 Quantization Algorithm

Conway and Sloane [17] developed algorithms for finding the closest lattice point of the n-dimensional lattices such as Zn, Dn and E8 lattice to an input vector x.

Finding the closest point of D4 lattice- For a given x =(x1, x2, x3, x4) vector following are the steps to find out the closest lattice points of D4 lattice.

Step1: f(x) = round(x).

Step 2: Check sum {f(x)}, if it is even number, f(x) will be the closest lattice point of x, else if sum is an odd number then do step 3.

Step 3: Calculate h(x). To calculate h(x) change one coordinate value of f(x), for which the difference between the original value of the coordinate of x and its round off value in f(x) is maximum.

For example if x1= (0.2, 1.8, -0.7, 0.4) and x2 = (0.4, 0.7,-1.1, 1.8) are two input vectors, then according to the algorithm

Step 1: f(x1) = (0, 2,-1, 0) and f(x2) = (0, 1, -1, 2)

Step 2: Sum{f(x1)} = 1 (odd number) and Sum{f(x2)} = 2 (even number), for x2 Sum{f(x2)}is an even number, so the closest lattice point will be f(x2) i.e. (0,1,-1,2), while for x1 Sum{f(x1)}is an odd number so step 3 is

Step 3: h(x1) = (0, 2,-1,1), sum{h(x1)}= 2 (even number), so the closest lattice point for x1 will be (0, 2,-1,1).

Finding the closest point of E8 lattice- Following are the steps to find out the closest point of E8 lattice to x. Given .

Step 1- Compute f(x) and g(x), where f(x) is closest integer to x and g(x) is same as f(x) except that the worst component of x, that furthest from an integer, is rounded the wrong way.

Step 2- Select whichever has an even sum of components, either f(x) or g(x), call the result z0

Step 3-Compute and , where and select whichever has an even sum of components, add and call the result z1.

Step 4-Calculate (x- z0)2 and (x- z1)2

Step 5- Compare the result. If (x- z0)2 < (x- z1)2 choose z0 as the closest point of x, else choose z1

2.5 Indexing the vectors in truncated lattices

When for a vector, lattice point is assigned, the next step is to assign index to that point. There are various indexing methods proposed for different lattice types and truncation in the literature. An indexing technique based on the notion of leader vector of a lattice was developed in [9] for and lattices. In [3] an enumeration based on signed leaders or generated signed leaders valid for a large class of lattices (, , and) is presented. A method for indexing a leader based on partition function having link with the number theory is described in [11], [12].

They used a partition function q(r, n) which not only gives the total number of leaders lying on a given hyper-pyramid but can also be used to provide unique indices for these leaders. For indexing of lattice quantized vectors product code indexing method is used to assign indexes to the quantized vectors. Index assigned by this method consists of a pair of prefix and suffix. Where prefix is the radius R or norm or energy of the vector, it can be easily encoded. Suffix is the index of the position of the vector on the shell of radius R. Indexing and coding of suffix is a difficult operation. Indexing of the suffixing can be done in two ways.

In first method shown in Table 5 all same norm vectors of D4 lattice and pyramidal truncation are arranged in lexicographical order and indexes are assigned to them. So in this case final index is (energy of the vector, index of the vector on the shell of same energy). While in the second method index of suffix is based on leader addressing, in this case index of suffix consists of leader index and rank of the vector in the class of equivalence shown in Table2.6

Table 2.5: Indexing of suffix based on enumerating lattice points

Prefix (norm or radius or energy)	Lattice code vector	Suffix (position of the vector on the shell of energy 2)	Index= (Prefix, Suffix)
2	2 0 0 0	1	(2,1)
	0 2 0 0	2	(2,2)
	0 0 2 0	3	(2,3)
	0 0 0 2	4	(2,4)
	1 1 0 0	5	(2,5)
	1 0 1 0	6	(2,6)
	1 0 0 1	7	(2,7)
	0 1 1 0	8	(2,8)
	0 1 0 1	9	(2,9)
	0 0 1 1	10	(2,10)
	1 0 0 -1	11	(2,11)
	1 0 -1 0	12	(2,12)
	1 -1 0 0	13	(2,13)
	0 1 0 -1	14	(2,14)
	0 1 -1 0	15	(2,15)
	0 0 1 -1	16	(2,16)
	0 0 -1 1	17	(2,17)
	0 -1 1 0	18	(2,18)
	0 -1 0 1	19	(2,19)
	-1 1 0 0	20	(2,20)
	-1 0 1 0	21	(2,21)
	-1 0 0 1	22	(2,22)
	0 0 0 -2	23	(2,23)
	0 0 -2 0	24	(2,24)
	0 -2 0 0	25	(2,25)
	-2 0 0 0	26	(2,26)
	0 0 -1 -1	27	(2,27)
	0 -1 0 -1	28	(2,28)
	0 -1 -1 0	29	(2,29)
	-1 0 0 -1	30	(2,30)
	-1 0 -1 0	31	(2,31)
	-1 -1 0 0	32	(2,32)

Table 2.6: Indexing of suffix based on leader indexing

Norm	Absolute Leaders	Signed Leaders	Index of Signed Leader	Class of equivalence	Vectors	Rank of the vector	Index of the lattice point
2	2 0 0 0	2 0 0 0	1	0 2 0 0 0 0 2 0 0 0 0 2	2 0 0 0 0 2 0 0 0 0 2 0 0 0 0 2	1 2 3 4	(2,1,1) (2,1,2) (2,1,3) (2,1,4)
	1 1 0 0	1 1 0 0	2	1 0 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 1	1 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 1	1 2 3 4 5 6	(2,2,1) (2,2,2) (2,2,3) (2,2,4) (2,2,5) (2,2,6)
		1 0 0 -1	3	1 0 -1 0 1 -1 0 0 0 1 0 -1 0 1 -1 0 0 0 1 -1 0 0 -1 1 0 -1 1 0 0 -1 0 1 -1 1 0 0 -1 0 1 0 -1 0 0 1	1 0 0 -1 1 0 -1 0 0 1 -1 0 1 0 0 -1 0 1 0 -1 1 0 -1 0 0 0 1 -1 0 0 -1 1 0 -1 1 0 0 -1 0 1 -1 1 0 0 -1 0 1 0 -1 0 0 1	1 2 3 4 5 6 7 8 9 10 11 12	(2,3,1) (2,3,2) (2,3,3) (2,3,4) (2,3,5) (2,3,6) (2,3,7) (2,3,8) (2,3,9) (2,3,10) (2,3,11) (2,3,12)
		0 0 0 -2	4	0 0 -2 0 0 -2 0 0 -2 0 0 0	0 0 0 -2 0 0 -2 0 0 -2 0 0 -2 0 0 0	1 2 3 4	(2,4,1) (2,4,2) (2,4,3) (2,4,4)
	0 0 -1 -1		5	0 -1 0 -1 0 -1 -1 0 -1 0 0 -1 -1 0 -1 0 -1 -1 0 0	0 0 -1 -1 0 -1 0 -1 1 0 -1 0 1 -1 0 0 -1 0 0 -1 0 -1 -1 0 0 -1 0 -1 -1 -1 0 0	1 2 3 4 5 6	(2,5,1) (2,5,2) (2,5,3) (2,5,4) (2,5,5) (2,5,6)

2.6 Coding of index

Entropy coding or lossless coding is traditionally the last stage in an image compression scheme. The run-length coding technique is very popular choice for lossless coding. [13] Reports an efficient entropy coding technique for sequences with significant runs of zeros. The scheme is used on test data compression for a system-on-chip design. [14] the scheme incorporates variable length coding and Golomb codes which provide a unique binary representation for run-length integer symbol with different lengths. Golomb coding algorithm contains tunable parameter M, run length N means count of continuous number of 0's followed by 1. The first step in the encoding procedure is to select the Golomb code parameter M i.e. group size. The value of M is taken in which the compression ratio is highest. Once the group size M is determined, the runs of zeros in precomputed test set are mapped to groups of size M (each group corresponding to a run length). The number of such groups is determined by the length of the longest run of zeros in the precomputed test set. The set of run lengths {0, 1, 2 . . ., m-1} forms group G1; the set {m, m+1, m+2 . . . , 2m-1}, G2 group; etc. In general, the set of run lengths {(k-1) m, (k-1) m+1, (k-1) m+2 . . ., km-1} comprises group Gk [15]. To each group Gk, we assign a group prefix of (k - 1) 0s followed by 1. We denote this by 0(k-1)1. If M is chosen to be a power of two, i.e., M = 2m, each group contains 2m members and a log2M-bit sequence (tail) uniquely identifies each member within the group. Thus, the final code word for a run length L that belongs to group Gk is composed of two parts, a group prefix and a tail. The prefix is 0(k-1)1 and the tail is a sequence of log2M bits. It can be easily shown that (k - 1) = (m mod M) i.e., k = (m mod M) + 1. The encoding process is illustrated in table 1 for M = 4. First group will consist of the run length {0,1,2,3}and second group will consist of the run length {4,5,6,7} and so forth. Group 1 will have a prefix {0}, group 2 will have prefix {10}, and so forth. Since the value of group size is 4, the length of tail is log24=2. For run length 0 the tail is represented by bits {00, the tail for run length 1 is represented by bits {10}, and so forth. Since the codeword is the combination of the group prefix and the tail, for run-length of 0 will have the codeword of {000}, the run length 1 will have codeword {001}, and so forth. If the input sequence has continuous 1's and 0's then Golomb coding is not a suitable method to code this type of sequence, because after coding number of bits get increased. So the solution is, two value Golomb coding (2-V Golomb coding) [16]. This method is same as Golomb coding with the difference that we calculate run length as count of continuous 0's or continuous 1's. Advantage of such modification in algorithm is that as both runs of 0's and 1's are considered, so large number of continuous ones may also be coded with few bits, which is not possible with Golomb coding.

Table 2.7: Golomb coding for M=4

Group	Run length	Group prefix (Quotient)	Tail (Remainder)	Codeword
G_1	0	1	00	100
	1		01	101
	2		10	110
	3		11	111
G_2	4	01	00	0100
	5		01	0101
	6		10	0110
	7		11	0110
G_3	8	001	00	00100
	9		01	00101
	10		10	00110
	11		11	00111
G_4	12	0001	00	000100
	13		01	000101
	14		10	000110
	15		11	000111
...	...	...	...	...

Another advantage is that in Golomb coding if the input sequence does not end with one we have to apply extra bit 1 at the end but we don't have to insert 1 at the end of sequence in two value Golomb coding. This modified algorithm is known as 2-V Golomb coding as both runs of 0's and 1's are considering. For example to represent 45 input bits shown as S, Golomb coding representation requires 64 bits while with 2-value Golomb coding only 33 bits are required. S_1 and S_2 show group of input bits in case of Golomb coding and 2-value golomb coding respectively. Input sequence
(S)={000000001000000000010000111111111111000000001 1 }
S_1= {000000001 00000000001 00001 1 1 1 1 1 1 1 1 1 1 1 1 00000001 1}
S_2= {00000000 1 0000000000 1 0000 111111111111 0000000 11}

Table 2.8: Comparison of Golomb coding and Two-value Golomb coding

(Golomb coding)									Total bits
Runlength	8	10	4	11 times 0	7	1	-	-	
Codeword	00100	00110	0100	11 times 000	0111	101			64
(2-value Golomb coding)									Total bits
Runlength	8	1	10	1	4	12	7	2	
Codeword	00100	101	00110	101	0100	000100	0111	110	33

III. RESULT

In this work, initially image coding using Lattice Vector Quantization (LVQ) with Generator Matrix (GM) and codebook is implemented. When implementation using codebook is done,two codebooks are constructed, one with 256 lattice points that are closest to (0,0,0,0) and another with 256 lattice points that are closest to (1,0,0,0). Energy for both the codes is calculated. When we compare the energy of both the codes we find that codes centered at a nonlattice point is lower energy code.

IV. CONCLUSION

In this work, different LVQ methods are applied for image compression on various standard test images and the outputs are analyzed. Lattice Vector Quantization performance depends on the length of the codebook. If the number of lattice points is less in a codebook, the time required to get quantized lattice point will be less but at the same time quality of reconstructed image will be degraded. If the size of the codebook is larger then the time required to search nearest vector will be more. So to avoid these situations, one solution is expansion of codebook when needed.

References

[1] M. Shnaider, "Lattice Vector Quantization for Wavelet based Image Coding", in Advances in Imaging and Electron Physics, Vol. 109.

[2] J. H. Conway and N.J.A. Sloane, "Voronoi Region of Lattices, Second Moments of Polytopes, and Quantization," IEEE transaction on Information Theory, Vol. IT-28, pp. 211-226, Mar.1982.

[3] Patrick Rault and Christine Guillemot, "Lattice vector quantization with reduced or without look-up table", Proc. SPIE Vol. 3309, pp. 851-862, Visual Communications and Image Processing '98

[4] Won-Ha Kim, Yu-Hen Hu and Truong Q. Nguyen, "Wavelet – Based Image Coder with Entropy Constrained Lattice Vector Quantizer (ECLVQ)" IEEE transactions on circuits and systems-II: Analog and Digital Signal Processing, 1998, Vol-45, No.8., pp. 1015-1030.

[5] Adriana Vasilache, "Entropic Encoding Of Lattice Codevectors Based on Product Code Indexing", 16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

[6] N. J. A. Sloane, "Tables of Sphere Packings and Spherical Codes," IEEE transactions on Information Theory, Vol. IT-27, No. 3, 1981.

[7] P. Sole, "Generalized Theta Functions for Lattice Vector Quantization," Proceedings of IEEE International Symposium on Information Theory, 1993

[8] Michel Barlaud, Patrick sole, Thierry Gaidon, Marc Antonini and Pierre Mathieu, "Pyramidal Lattice Vector Quantization for Multiscale Image Coding," IEEE Transaction of Image Processing, Vol.3, No. 4, pp. 367-381, July 1994 .

[9] J. Moureaux, P. Loyer, and M. Antonini, "Low Complexity Indexing Method for Z^n and D^n Lattice Quantizers," IEEE Trans. Commun., Vol. 46, No. 12, pp. 1602–1609, December 1998.

[10] A. Vasilache, M. Vasilache,. & I. Tabus, " Predictive Multiple-Scale Lattice VQ for LSF Quantization," Proceedings of ICASSP 1999, pp. 657–660, 1999

[11] L. H. Fonteles and M. Antonini , "High Dimension Lattice Vector Quantizer Design for Generalized Gaussian Distributions," IEEE International Conference (ICIP) 2007

[12] L. H. Fonteles and M. Antonini, "Indexing Z^n Lattice Vectors for Generalized

Gaussian Distributions," in Proc. of IEEE ISIT, pp. 241-245, June 2007.

[13] A. Chandra and K. Chakrabarty, "System-on-a-chip Test Data Compression and Decompression Architectures based on Golomb Codes," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 20, No. 3, pp. 355–368, 2001.

[14] S. W. Golomb, "Run-length Encodings," IEEE Transactions on Information Theory, Vol. 12, No. 3, pp. 399–401, 1966.

[15] M.F.M. Salleh and J. Soraghan, "A New Multistage Lattice Vector Quantization with Adaptive Subband Thresholding for Image Compression," EURASIP Journal on Advances in Signal Processing, Vol. 2007, article ID 92928, 11 pages, 2007.

[16] Po-Chang Tsai, Sying-Jyan Wang, Ching-Hung lin, Tung-Hua Yeh, "Test data =compression for Minimum Test ==Application Time", Journal of Information Science and Engineering 23, 1901-1909 (2007).

[17] J. H. Conway and N. J. A. Sloane, "Fast Quantizing and Decoding Algorithms for Lattice Quantizers and Codes," IEEE transaction on Information Theory, Vol. IT-28, pp. 227-232, Mar.1982.