International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 4, Issue 7, pp 1-8, July-2020

https://doi.org/10.47001/IRJIET/2020.407001

Finite-State Vector Quantization Techniques for Image Compression

***Srijati Agrawal**

*Electronics and Computer Engineering, Trinity College Dublin, Ireland

Abstract **- Different techniques of Finite-State Vector Quantization, in the image coding framework are investigated in this paper. Introduced in this paper are two FSVQ designs, conventional vector quantization and side-match vector quantization. Convention Vector Quantization divides an image into K-dimensional blocks where each vector x is mapped to its corresponding code vector from a codebook of size N. It exploits the correlation between adjacent pixels within a block of pixels. The encoder of a vector quantizer uses the previously encoded blocks to make a selection from a family of codebooks. Although vector quantization provides an acceptable visual quality of the compressed image, it does it at the cost of increased bit rate. The Side Match Vector Quantization reduces the bit rate required for storage and transmission of the image and provides a good visual quality image. SMVQ requires creating its own state codebook for each block for encoding as well as decoding. SMVQ takes advantage of the spatial contiguity of pixel vectors by exploiting the correlations of the nearest neighboring blocks. They try to minimize the granular noise that causes the annoying effect of visible pixel block boundaries in conventional VQ. Since the conventional method for the generation of state codebook for SMVQ is time consuming, the method of rapid generation of state codebook proposes a fast method for codebook generation.**

In this paper, different image compression techniques of vector quantization, side-match vector quantization and rapid generation of state codebook method will be implemented to evaluate the best possible method. Although each technique is an improvement over the other, the proposed method for rapid generation of state codebook is faster than the others and without any loss of perpetual visual quality. The Linde-Buzo-Gray algorithm, also known as LBG algorithm, is used for the generation of codebooks.

Keywords: vector quantization, side-match vector quantization, LBG algorithm, Rapid generation of state codebook, code book, state-codebook, codewords.

I. INTRODUCTION

Images are an important and popular form of media. Image compression is the process of compressing digital images to reduce the cost of their storage and transmission. Image compression techniques can be classified into two categories: lossless and lossy image compression techniques. Lossless image compression techniques involve no loss of information. In lossy compression techniques, some loss of information is involved, and data that have been compressed using lossy techniques generally cannot be recovered or reconstructed exactly. Because of its high compression rate, lossy image compression techniques have been widely used in image compression. Among the lossy data compression techniques, vector quantization (VQ) is one of the most popular data compression techniques.

Vector Quantization is a simple technique used for compression in the field of image and video processing. Side Match Vector Quantization is also a technique of vector quantization which exploits the correlations between the borders of two neighboring blocks to reduce the bitrate of compressed image. The image to be encoded is first partitioned into non-overlapping rectangular blocks of size 4x4(16-dimentional). VQ can be defined as a mapping from k-dimentional Eucledian space into a finite subset C of R^k, where C=$\{c_i \mid i= 1,2,\ldots\ldots,N\}$. Here, C is called the codebook of the vector quantizer, c_i is the codeword or code vector and N is the size of the codebook.

In vector quantization, the coder has two parts: an encoder and a decoder. The encoder assigns each input vector x, to an index i, in the codebook C. The decoder finds the transmitted index i. The distortion is calculated by finding the Eucledian distance between each pixel in x and corresponding pixel in ci and summing over the square of all these distances. The formula is as follows:

$$d(x,y_i) = \|x\text{-}c_i\|^2 = \sum_{j=0}^{k-1}\left(x_j - c_{ij}\right)^2$$

A training set (TS) is used to generate a variety of different sized codebooks. It is a large array of vectors. Images generally used for creating the training set are as follows:

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 4, Issue 7, pp 1-8, July-2020
https://doi.org/10.47001/IRJIET/2020.407001

Fig 1(a): Bar

Fig 1(b): Boat

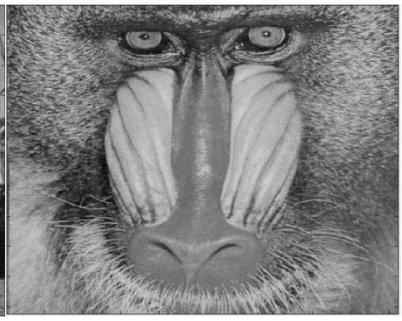

Fig 1(c): Mandrill

Fig 1(d): Moffett

Fig 1(e): Sailboat

Fig 1(f): Peppers

Fig 1(g): Tiffany

Fig 1(h): Airplane

1.1 LBG Algorithm

The Linde-Buzo-Gray algorithm, commonly known as LBG algorithm was proposed by Y. Linde, A. Buzo and R.M Gray in 1980. This method is commonly used for the generation of codebook with minimum error. LBG algorithm iteratively searches clusters in the training data. The cluster centroids are used as codebook vectors, while the codewords for them are selected arbitrarily.

1.1.1 Codebook Generation

The codebook can be generated by using a training set of images. The initial codebook is generated by selecting a certain set of code-vectors. For example, to create a codebook C, of size N, number of code-vectors to be initialized must be N as well. The basic method to do that is to select every nth vector from the training set, where n depends on the size of the codebook. For example, if N is 256, then every nth vector is the 32nd vector in the training set. Each of these vectors is used to design the initial codebook.

1.1.2 Clustering and centroid calculation

After an initial codebook is generated, a step by step selection process is applied to create clusters. For each training set vector t_j, the best matching vector c_i, is found. t_j is mapped to its corresponding c_i and the process is repeated for all training set vectors. 'Similar' vectors from the training set will gather under same cluster. Then the centroid of the cluster is calculated. The centroid then takes the place of the current code-vector and the clusters are created. The distortion between each c_i and t_j is calculated and added to the total distortion, T_{dist} of that cluster.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 4, Issue 7, pp 1-8, July-2020

https://doi.org/10.47001/IRJIET/2020.407001

1.1.3 Codebook Evaluation

To ensure the formation of an optimal codebook, fractional improvement in the reduction of distortion between the previous and new codebooks is measured. This improvement is the fractional drop in the total distortion between current code-vector and previous code-vector. It is calculated using following equation:

$$\text{Fractional Distortion Drop} = \frac{D_{m-1} - D_m}{D_m}$$

1.2 Side-Match Vector Quantization

Side-Match Vector Quantization (SMVQ) is a typical class of vector quantization and has been widely used in the field of data hiding [10]. SMVQ exploits the correlations between the borders of two neighboring blocks to reduce the bitrate of the compressed image [1][10]. It employs the previous coded blocks to help predict the current block so that the visible boundaries can be reduced.

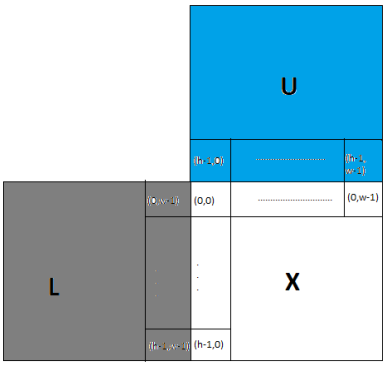

Figure 2: Relationship of neighboring pixel blocks in SMVQ

The state-codebook is a subset of main codebook which contains the best matching code-vectors or code-words from the main codebook C. For the given image, the topmost row and leftmost column are encoded in advance using the ordinary vector quantization with the main codebook. To generate a state codebook, side-match distortion is calculated between the border vector and side vector. The distortion is calculated by finding the Eucledian distance between side vector and border vector. The formula is as follows [11]:

$$d_i = \sqrt{\sum_{k=0}^{b_w + b_h - 2} [\, v_x(k) - v_{m_i}(k)]^2} \qquad (1)$$

The state codebook is generated by choosing the best matching code vectors that have the minimum distortion value. The code vectors obtained will be stored in an ascending order according to side-match distortion along with their index numbers. The first Ns smallest distortion codewords are selected to be used as the state codebook.

1.3 Rapid Generation of the State Codebook

The state codebook of size Ns is initialized and then optimized using the LBG algorithm. The generation of the state codebook can be done by taking every n^{th}vector from the training set. For example, in this paper, a main codebook of size 256 was generated from the training set using LBG algorithm. To create a state codebook of size 32 from a main codebook of size 256, every 32ndvector is taken from it. In this way, the state codebook also known as sub-codebook of size 32 was generated having 8 code-vectors in it ($\frac{256}{32} = 8$).

Each state codebook vector consists of a border vector of the codebook pixel block. The border vector is calculated from the top row and leftmost column of the current vector from the main codebook.

The 4 pixels above and the 4 pixels to the left of the current block are encoded using the ordinary vector quantization technique and border vector is generated. Using the border vector, map each of the side-vector to its neighbor in the main codebook and find the best- matching side vector. After repeating the process for all the code-vectors in the main codebook, a list of all the best-matching code-vectors is obtained.

To encode the image, the list of indices associated with the best-matching code-vectors is analysed. When the initial mapping of the codevectors is done, different amount of main codebook vectors get mapped forming clusters. Each cluster has a different number of side code-vectors mapped to it. The maximum and minimum number of mapped side code-vectors gave a general idea of the range of the codebook.

II. RESULTS AND DISCUSSION

For Vector Quantization, different codebooks of size 32, 64, 128, 256, 512 and 1024 were generated using the original training set with 8192 vectors and were trained using the LBG algorithm described previously.

The results are evaluated on the factors such as PSNR, bitrate and computation complexity.

$$\text{PSNR} = 10 \times \log_2 (255^2 / \text{MSE}) \qquad (2)$$

$$\text{MSE} = \frac{1}{K} \sum_{i=1}^{K} (x_i - \hat{x}_i)^2 \qquad (3)$$

$$\text{bpp} = \frac{\#bits}{\#pixels} \qquad (4)$$

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 4, Issue 7, pp 1-8, July-2020

https://doi.org/10.47001/IRJIET/2020.407001

The compression rate is calculated by comparing the number of bits before and after compression. The formula is given as follows [11]:

$$\text{Compression Rate (\%)} = (1 - N_b'/N_b) \times 100. \quad (5)$$

The computational complexity of the program is evaluated by measuring the full program run time and encoding time. It is measured in seconds. The 'timeit' function in MATLAB was used for this purpose.

The image Lena was compressed using ordinary vector quantization. Below is a table showing the test results for ordinary VQ. The results are evaluated on the factors such as PSNR, bitrate and computation complexity.

Table 1: VQ testing results on Lena

N	PSNR(db)	Encoding Time(s)	Bit Rate (bpp)
32	23.0430	0.03113	0.3125
64	26.4971	0.06234	0.375
128	28.5123	0.11924	0.438
256	29.4936	0.22037	0.5
512	30.1542	0.45326	0.563
1024	31.2511	0.87443	0.635

The figure below shows a plot of PSNR vs Codebook size for ordinary Vector quantization.

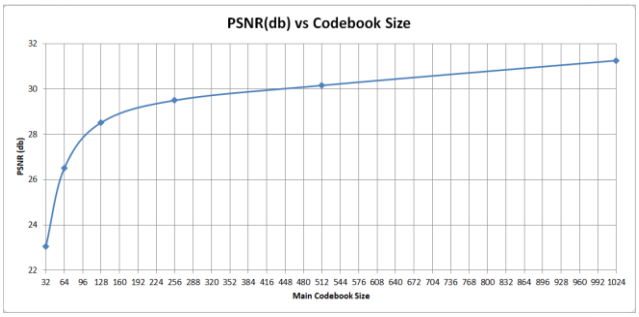

Figure 3: Plot of PSNR against Codebook size for VQ

The figure below shows Lena encoded with a codebook of size 32

Figure 4: Lena compresses using a codebook of size 32

2.1 VQ and SMVQ

SMVQ aims to reduce the bitrate while still maintaining good quality of the image. In this paper, the image, Lena was chosen to carry out the tests. The state codebooks of size 16, 32, 128, 256 and 512 were generated from main codebooks of size 256, 512 and 1024.

They were trained using LBG algorithm and the state codebooks were generated using conventional side-match vector quantization. The PSNR values are used to determine the quality of the compressed image. The compression rates were also calculated.

The table below shows the results of SMVQ for a codebook of size 256 with state codebooks of different sizes.

Table 2: SMVQ testing results for a codebook of size 256

State codebook size, Ns	Compression Rate (%)	PSNR(db)	Time(ms)
16	98.29	25.9857	684
32	97.45	28.8714	701
64	97.42	29.0846	725
128	96.54	29.9534	766

The table below shows the results of SMVQ for a codebook of size 512 with state codebooks of different sizes.

Table 3: SMVQ testing results for a codebook of size 512

State codebook size, Ns	Compression Rate (%)	PSNR(db)	Time(ms)
16	98.29	25.1754	1471
32	97.65	27.3386	1527
64	97.64	28.7921	1559
128	96.74	29.4835	1593
256	96.71	30.5145	1646

The table below shows the results of SMVQ for a codebook of size 1024 with state codebooks of different sizes.

Table 4: SMVQ testing results for a codebook of size 1024

State codebook size, Ns	Compression Rate (%)	PSNR(db)	Time(ms)
16	98.27	24.4331	3246
32	97.65	26.6644	3306
64	97.64	28.1830	3397
128	96.74	29.2074	3428
256	96.74	30.6983	3451
512	96.11	31.2556	3701

The plot below shows the PSNR values in decibels for state codebooks of different sizes created for main codebooks of size 256, 512 and 1024.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 4, Issue 7, pp 1-8, July-2020

https://doi.org/10.47001/IRJIET/2020.407001

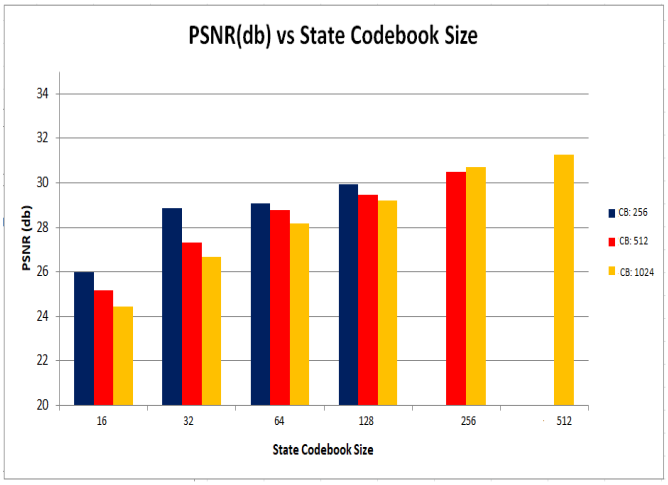

Figure 5: Plot of PSNR against State codebook size

The bar chart above depicts how the change in the size of the state codebook affects the PSNR values. Each block of bar corresponds to the size of the state codebook and each color corresponds to the size of the main codebook.

The table below shows the comparison between bitrates for main codebooks of size 256, 512 and 1024 for different state codebooks.

Table 5: Bitrate values for SMVQ for different codebooks and state codebooks

Ns / N	256	512	1024
16	0.1923	0.1933	0.1943
32	0.2538	0.2548	0.2558
64	0.3154	0.3163	0.3174
128	0.3769	0.3779	0.3789
256		0.4394	0.4404
512			0.4975

Below is a plot showing the distribution of bitrates for different state codebooks.

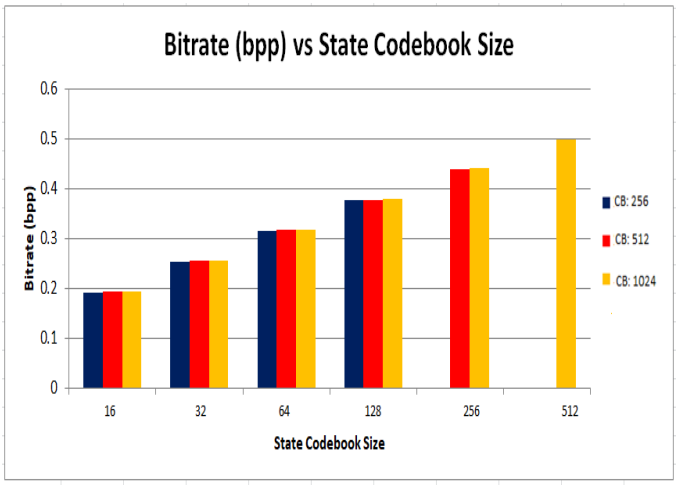

Figure 6: Plot of Bitrate against State codebook size

The bar chart above depicts how the increase in the size of the state codebook affects the bitrate.

Each block of bar corresponds to the size of the state codebook and each color corresponds to the size of the main codebook.

This shows that as the size of the state codebook increases, the bitrate also increases. This is because bitrate is the maximum number of bits required to transmit the largest index. When the tests are run using main codebooks of size 512 and 1024, it was noted that the bitrate never reaches a value above 0.5 which can be verified from table 5, as compared to VQ where the highest bitrates are 0.563 and 0.635 as can be seen from Table 1.

The Figure 7 below shows original Lena and the figure 8 shows Lena compressed using ordinary VQ using a codebook of size 256.

Figure 7: Original Lena

Figure 8: Lena Encoded

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 4, Issue 7, pp 1-8, July-2020

https://doi.org/10.47001/IRJIET/2020.407001

The change in PSNR values between VQ and SMVQ indicates that the image quality in SMVQ increases when the codebook size increases. This can be seen from the graph below.

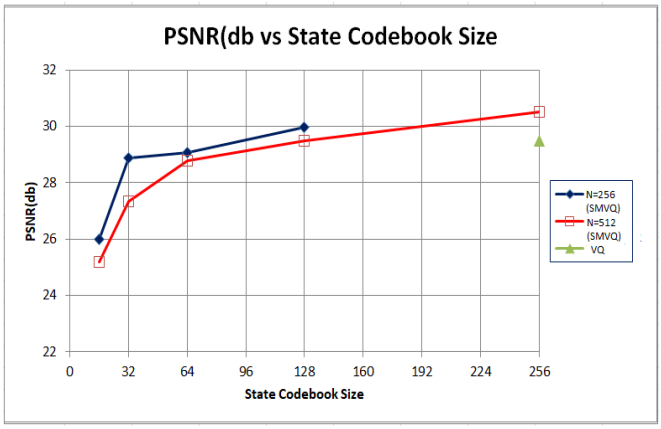

Figure 9: Plot of PSNR against the state codebook size of 256 and 512 for SMVQ

The green dot on the graph displays the PSNR of compressed image using VQ for a codebook of size 256.

2.2 Rapid Generation of the State Codebook

The main aim of RGSC is to speed up the generation of the state codebook and to reduce the overall computational complexity.

To check the performance of this method, tests were run on an image of size 512x512 called Lena. The compression rates, PSNR values processing times for the proposed method were computed for main codebooks of size 256, 512 and 1024. The tables below show the results obtained for RGSC.

Table 6: RGSC testing results for a codebook of size 256

State codebook size, Ns	Compression Rate (%)	PSNR(db)	Time(ms)
8	98.20	24.6812	22
16	97.55	26.6727	23
32	97.33	27.4592	25
64	97.19	28.5124	41
128	96.72	29.7710	68

Table 7: RGSC testing results for a codebook of size 512

State codebook size, Ns	Compression Rate (%)	PSNR(db)	Time(ms)
8	98.09	24.7892	32
16	97.82	26.0921	23
32	97.32	27.3655	43
64	97.23	28.4210	56
128	96.69	29.5282	69
256	96.32	30.4501	129

Table 8: RGSC testing results for a codebook of size 1024

State codebook size, Ns	Compression Rate (%)	PSNR(db)	Time(ms)
8	98.01	23.6098	57
16	97.56	26.0172	59
32	97.32	27.4815	65
64	96.89	28.6211	87
128	96.69	29.4965	98
256	96.25	30.1533	161
512	96.03	31.1596	261

From above tables for compression ratio of SMVQ and tables 2, 3, 4 for compression ratio of RGSC for main codebooks of 256, 512 and 1024 respectively,

It can be stated that there no significant change in the compression ratio for these two methods for state codebooks of given size and it does not affect the quality of image to a significant amount.

Below is a plot of the PSNR values for RGSC for different state codebooks.

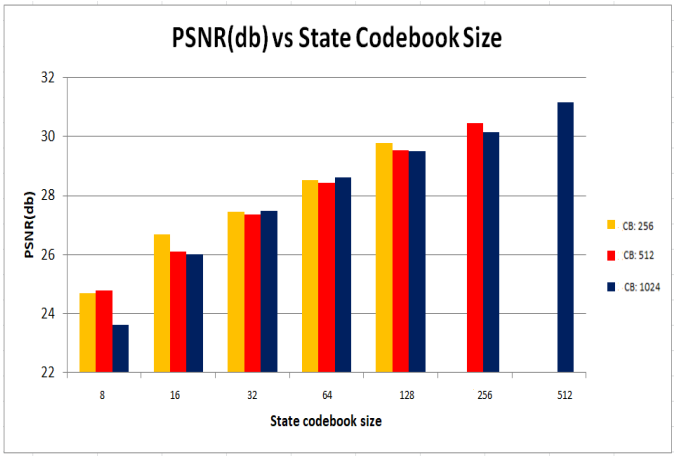

Figure 10: Plot of PSNR against State codebook size

From the table above it can be stated that there is considerably small difference between the PSNR values for state codebooks of different sizes. However, the codebook for size 1024 provides the highest PSNR value of 31.1596 decibels which results in the image of better visual quality.

III. CONCLUSION

The objective of this paper was to implement and evaluate different vector quantization techniques to obtain an image with good visual quality and reduced bitrate. In this paper, the ordinary vector quantization technique was implemented which provided an image with acceptable visual quality but an increased bitrate. To reduce the bitrate, side match vector

ISSN (online): 2581-3048
Volume 4, Issue 7, pp 1-8, July-2020
https://doi.org/10.47001/IRJIET/2020.407001

quantization was used for image compression. SMVQ was implemented by generation of state codebooks for main codebooks of different sizes. SMVQ had the advantage of two-dimentional spatial contiguity of pixel blocks as it exploited the correlations between the nearest neighbors in grey level images. After implementing SMVQ, it was noted that the image reconstructed using this technique is of good visual quality with a reduced bitrate.

However, the good visual quality of the compressed image is dependent on the size of the state codebook. It was noted that the image quality improved after a state codebook of size 128 was used. As the size of state codebook increased, the image quality also improved. But with the increase in the size of state codebook, the computation complexity of SMVQ also increased. SMVQ generates good quality compressed image but at the cost of time.

To generate the codebook rapidly, the method of Rapid Generation of State Codebook was implemented. This method was based on the generation of sub codebook and clustering the main codewords. When implemented, the proposed method of RGSC outperformed the side match vector quantization method in terms of the state codebook generation speed and perpetual visual quality of the image. The computational complexity for RGSC was lesser than that of the ordinary VQ and conventional SMVQ. The image generated using rapid state codebook generation method was of better visual quality and the computational complexity was vastly improved compared to that of VQ and SMVQ.

REFERENCES

[1] Taejeong Kim, "Side Match and Overlap Match Vector Quantizers for Images", *IEEE Transactions on Image Processing*, Vol. 1, No.2, April 1992.

[2] Ruey-Feng Chang and Wen-Tsuen Chen, "Image coding using Variable-Rate Side-Match Finite-State Vector Quantization", *IEEE Transactions on Image Processing*, Vol. 2, No.1, January 1993.

[3] Chaur-Heh Hsieh, Kuo-Chiang and Jin-Sen Shue. "Image Compression using Finite-State Vector Quantization with Derailment Compensation", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No.5, October 1993.

[4] Ruey-Feng Chang and Wei-Ming Chen, "Adaptive Edge-Based Side-Match Finite-State Classified Vector Quantization with Quadtree Map", *IEEE Transactions on Image Processing*, Vol.5, No.2, February 1996.

[5] Tung-Shou and Chin-Chen Chang, "A New Image Coding Algorithm Using Variable-Rate Side-Match Finite-State Vector Quantization", *IEEE Transactions on Image Processing*, Vol.6, No.8, August 1997.

[6] Ruey-Feng Chang and Wen-JiaKuo, "Image Coding Using Two-Pass Side-Match Finite-State Vector Quantization", *Journal of Inforamtion Science and Engineering*, 15, 41-51 (1999).

[7] Jyi-Chang Tsai, Chaur-Heh Hsieh and Te-Cheng Hsu, "A New Dynamic Finite-State Vector Quantization Algorithm for Image Compression", *IEEE Transactions on Image Processing*, Vol.9, No.11, November 2000.

[8] Shiueng Bien Yang and Lin Yu Tseng, "Smooth Side-Match Classified Vector Quantizer with Variable Block Size", *IEEE Transactions on Image Processing*, Vol.10, No.5, May 2001.

[9] Linde, Y., Buzo,A., Gray, R.M., "An Algorithm for Vector Quantizer Design", *IEEE Transactions on Communication*, Vol.Com-28, No.1 , January 1980.

[10] Xiaoxiao Ma, Zhibin Pan, Sen Hu and Lingfei Wang, "Enhanced Side-Match Vector Quantization based on constructing Complementary State Codebook", *IET Image Processing*, Vol 10, August,2014.

[11] Hanhoon Park and Jong-II Park, "Rapid Generation of the State Codebook in Side Match Vector Quantization", *IEICE Trans. Inf. & Syst.*, Vol.E100-D, No.8, August 2017.

[12] K. Somasundaram and S.Domnic, "Modified Vector Quantization Method for Image Compression", - *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, Vol.2, No.7, 2008.

[13] Hsien-Chung Wei, Pao-Chin Tsai and Jia-Shung Wang, "Three-Sided Side Match Finite State Vector Quantization", *Department of Computer Science*, National Tsing Hua University.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 4, Issue 7, pp 1-8, July-2020

https://doi.org/10.47001/IRJIET/2020.407001

Citation of this Article:

Srijati Agrawal, "Finite-State Vector Quantization Techniques for Image Compression" Published in *International Research Journal of Innovations in Engineering and Technology - IRJIET*, Volume 4, Issue 7, pp 1-8, July 2020. https://doi.org/10.47001/IRJIET/2020.407001

****

