# SPARSE BASED FINGERPRINT COMPRESSION AND COMPARISON WITH SPIHT AND WSQ

[1]Kalandar Shafi, [2]Maya V. Karki
[1]Student, [2]Associate Professor
M. S. Ramaiah Institute of Technology,Bangalore, India.
Email: [1]shafisera@gmail.com, [2]mayavkarki@msrit.edu

**Abstract— The increase in collection of fingerprints created the problem of storage and transmission. In order to reduce the storage and transmission bandwidth, compression techniques are needed. Many image compression techniques are available at present. This paper introduces sparse based algorithm to compress fingerprint. Using an adapted dictionary that contains prototype atoms of patch, fingerprint can be described as a sparse linier combination of these atoms. Sparse coefficients that represents given fingerprints are quantized and entropy encoded. The algorithm is tested on fingerprint databases FVC 2000, FVC 2002, FVC 2004 and our own database. The result of proposed algorithm is compared with WSQ and SPIHT algorithms. The experimental results show that the fingerprint compression using proposed algorithm gives better result compared to WSQ and SPIHT in most of the cases. The experiment shows the compression ratio of 35:1 with maximum PSNR of 30.32dB for FVC2004 database.**

**Keywords— Compression Ratio, Fingerprint, PSNR, Sparse, SPIHT, WSQ.**

## I. INTRODUCTION

Fingerprints are the ridge and curve patterns on the tip of finger [1], [2]. It plays important role in legal matters such as authentication of person, investigation of crime and many other security applications [3], [4]. Among many biometrics fingerprint is one of the matured technique because of their immutability and individuality [5]. Immutability refers to unchanging character of fingerprint pattern before birth till decomposition after death and Individuality refers to uniqueness of patterns across the individuals. The increase in fingerprint collection created the problem for storage and transmission. Although there are many image compression techniques are available, there is a need for developing faster, robust and less complexity algorithm for fingerprint compression.

Difficulty in developing fingerprint compression algorithm is need for preserving recognition parameters used for identification after compression.

Compression techniques are classified into lossless and lossy compression. Lossless compression techniques are able to reconstruct the image exactly same as original, but it gives less compression ratio. Lossless compression techniques are able to give higher compression ratio, but it loses some information of the original image.

Generally lossy image compression techniques includes transforming an image to other domain by using Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT), quantizing transformed coefficients and entropy encoded.

DCT based compression techniques includes dividing image into 8×8 block, DCT is applied to

each block to get coefficients, these are quantized and then entropy encoded. DCT compression techniques are used in JPEG [6]. JPEG compression is simple, but when the compression ratio is high it will not be able reconstruct the image efficiently.

The DWT based algorithm includes applying DWT to the normalized image to get coefficients, these are quantized and then entropy encoded. DWT based compression technique is used in JPEG2000 [7].

There are other DWT based algorithms such as Set Partitioning in Hierarchical Trees (SPIHT) algorithm [8] and FBI standard WSQ algorithm [9],[10].These to algorithms are used along with the proposed algorithm to compare the result.

This paper also gives fingerprint compression based on sparse representation [11]. It includes construction of dictionary, each column of dictionary is known as atom. Fingerprint are divided into small blocks called patches, whose dimension is equal to atom size. Coefficients are obtained by using method of sparse representation. These coefficients are quantized and entropy encoded.

Performance is measured using Peak Signal to Noise Ratio (PSNR) and Compression Ratio. High PSNR indicates that the reconstructed image retained more components after compression. Compression Ratio indicates number of bits required to represent fingerprint after compression as that of original.

## II. SPIHT algorithm

The SPIHT algorithm is a more efficient algorithm which was presented by Shapiro [12]. The SPHIT algorithm includes following steps: initialization, sorting pass, refinement pass, quantization and step update pass.

After applying wavelet transform to an image, the SPIHT algorithm divides the coefficients into significant and insignificant partitions based on the following function:

$$S_n(T) = \begin{cases} 1, & max_{(i,j)\epsilon T}\{|c_{i,j}|\} \geq 2^n \\ 0, & otherwise \end{cases} \dots (1)$$

Where $S_n(T)$ is the significance of a set of coordinates T, and $c_{i,j}$ is the coefficient value at coordinates (i, j). There are two passes in the algorithm sorting pass and the refinement pass. The SPIHT encoding process consist of three lists:

LIP (List of Insignificant Pixels) – it contains individual coefficients that have values smaller than thresholds.

LIS (List of Insignificant Sets) – it consist of group of coefficients that are defined by tree structures and are found to have magnitudes less than the threshold.

LSP (List of Significant Pixels) – it consist of coefficients larger than the threshold.

**Initialization:** In this stage threshold N, LSP, LIP, LIS are initialized. Threshold is initialized as given in equation (2). Therefore LSP becomes empty, LIP consist of pixels less than the threshold N, and LIS consist of set of pixels less than threshold N. After initialization algorithm iteratively repeats by decreasing threshold N as N/2.

$$n_{max} = [log_2(max_{i,j}\{|c_{i,j}|\})] \dots\dots\dots (2)$$

**Sorting pass:** The purpose of sorting pass is to manipulate the contents of LIP, LSP, LIS, so that they are correct with respect to the current threshold. During sorting pass, coordinates of the coefficients remain in LIP are tested for significant. The result is sent to output and out of it the significant will be transferred to the LSP as well as sending sign bit to the output. Sets in LIS also tested for significant, if that found to be significant, it will be removed and partitioned into subsets. Subsets with only one coefficient and found to be significant, will be eliminated and divided into subsets. Subsets having only one coefficient and found to be significant will be inserted to the LSP, otherwise they will be inserted to the LIP.

**Refinement pass:** The refinement pass follows the sorting pass and gives out the bit corresponding to the current value of threshold for each of pixels in the LSP which were not

added in the immediately previous sorting pass. In this pass, the $n^{th}$ MSB of the coefficients in the LSP is passed to the output.

In Quantization and Step update pass, the value of threshold N decremented as N/2. The sorting and refinement pass is been repeated until threshold N reaches to zero and all nodes in the LSP have all their bits output. The latter case will give an almost exact reconstruction since all the coefficients have been processed completely.

The bit rate can be controlled exactly in the SPIHT algorithm as the output produced is in single bits and the algorithm can be finished at any time. The decoding process follows the encoding exactly and is almost symmetrical in terms of processing time.

## III. WSQ ALGORITHM

A simplified block diagram of WSQ is given in figure 1. The algorithm consist of three main steps: decomposition of original fingerprint by applying discrete wavelet transform, these wavelet coefficients are quantized using uniform scalar quantization, and these quantized coefficients are entropy encoded.

In WSQ encoder, the original fingerprint is decomposed into 64 subbands of wavelet coefficients by applying discrete wavelet transform with level 5 as shown in figure 2. These subbands are quantized using adaptive uniform scalar quantization technique. Quantization coefficient of $k^{th}$subband $a_k(m,n)$ is given by the equation (3).

$$p_k(m,n) = \begin{cases} \left\lfloor \frac{a_k(m,n)-(0.6\,Q_k)}{Q_k} \right\rfloor + 1 \;\; ; \; a_k(m,n) > 0.6\,Q_k \\ 0 \; ; \; -0.6\,Q_k \le a_k(m,n) \le 0.6\,Q_k \\ \left\lceil \frac{a_k(m,n)+(0.6\,Q_k)}{Q_k} \right\rceil - 1 ; a_k(m,n) < 0.6\,Q_k \end{cases} \quad (3)$$

At the decoder, the de-quantization of a quantized coefficients are computed using equation (4).

$$\hat{a}a_k = \begin{cases} (p_k(m,n) - C)Q_k + (0.6\,Q_k); p_k(m,n) > 0 \\ 0 \qquad\qquad\qquad\qquad ; p_k(m,n) = 0 \\ (p_k(m,n) + C)Q_k - (0.6\,Q_k); p_k(m,n) < 0 \end{cases} \quad \ldots.(4)$$

Where $a_k(m,n)$ is wavelet coefficient of $k^{th}$ subband, $Q_k$ represents Quantization table value of $k^{th}$ subband, C determines quantization bin width,

$$Q_k = \begin{cases} 1/q, & k = 1\ to\ 4 \\ 10/(qA_k \log_e(\sigma_k^2)), & k = 5\ to\ 60\ and\ \sigma_k^2 \ge 1.01 \\ 0, & k = 61\ t0\ 64\ or\ \sigma_k^2 < 1.01 \end{cases} \quad (5)$$

Where $\sigma_k^2$ is variance of $k^{th}$subband and q can be set to get prespecified compression ratio and $A_k$ given by equation (6).

Quantized coefficients are entropy encoded using run length encoding method followed by Huffman encoding. Run length encoding is done according to the conditions given in table 2.

$$A_k = \begin{cases} 1.32, \; k = 53{,}57 \\ 1.08, \; k = 54{,}59 \\ 1.42, \; k = 55{,}58 \\ 1.08, \; k = 56{,}60 \\ 1.00, \; otherwise \end{cases} \quad \ldots\ldots (6)$$

Coefficient values are mapped to the symbols according to Table 2. After run length coding the symbols are grouped into three and Huffman encoding is applied to each group separately.

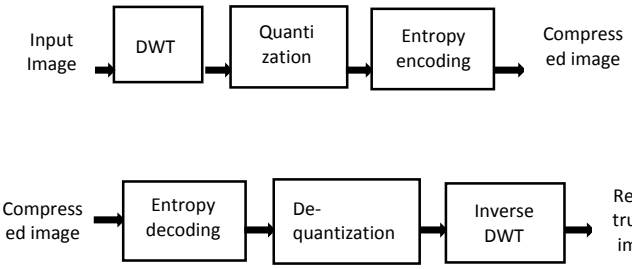


Figure 1. Simplified WSQ encoder and decoder block diagram

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 3 | 2 4 | 5 | 8 | 9 | 20 | 21 | 24 | 25 | 53 | 54 |
| 6 | | 7 | 10 | 11 | 22 | 23 | 26 | 27 | | |
| 12 | | 13 | 16 | 17 | 28 | 29 | 32 | 33 | | |
| 14 | | 15 | 18 | 19 | 30 | 31 | 34 | 35 | | |
| 36 | | 37 | 40 | 41 | 52 | | | | 55 | 56 |
| 38 | | 39 | 42 | 43 | | | | | | |
| 44 | | 45 | 48 | 49 | | | | | | |
| 46 | | 47 | 50 | 51 | | | | | | |
| 57 | | | | | 58 | | | | 61 | 62 |

| | | | |
|---|---|---|---|
| 59 | 60 | 63 | 64 |

Figure 2. WSQ standard DWT decomposed sub bands

## IV. SPARSE ALGORITHM

Sparse algorithm includes construction of the dictionary, sparse encoding, quantization and entropy coding. The simplified flow diagram of sparse algorithm is shown Figure 3.

A. Construction of dictionary

Fingerprint image is divided into patches with equal size. Initially dictionary is empty and first patch is added to the dictionary. Next patch is tested whether it is similar to the patch present in the dictionary by using similarity measure equation (7).

$$S(P1,P2) = min \left\| \frac{P1}{\|P1\|_F^2} - t * \frac{P2}{\|P2\|_F^2} \right\|_F^2$$

….. (7)

Where $\|\cdot\|_F^2$ represents Frobenius norm, P1, P2 indicates patches used for similarity measure and t is a scaling factor.

| Symbol | Value |
|---|---|
| 1 | Zero run of length 1 |
| 2 | Zero run of length 2 |
| 3 | Zero run of length 3 |
| . | . |
| . | . |
| 100 | Zero run of length 100 |
| 101 | Escape for positive 8 bit coefficient |
| 102 | Escape for negative 8 bit coefficient |
| 103 | Escape for positive 16 bit |
| 104 | coefficient |
| 105 | Escape for negative 16 bit |
| 106 | coefficient |
| 107 | Escape for 8 bit zero run |
| 108 | Escape for 16 bit zero run |
| . | Coefficient value -73 |
| . | Coefficient value -72 |
| 179 | . |
| 180 | . |
| 181 | Coefficient value -1 |
| . | (same as symbol 1) |
| . | Coefficient value 1 |
| 253 | . |
| 254 | . |
| | Coefficient value 73 |
| | Coefficient value 74 |

Table 2: Huffman coding model [13]

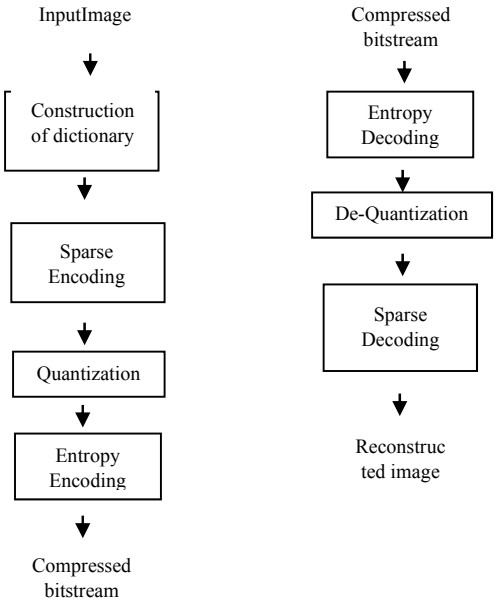


Figure 3. Simplified block diagram of sparse algorithm

If patch under test is similar to any patches in the dictionary, then atom number corresponding to that patch is stored. Otherwise the patch under test is added to the dictionary. In this work, dictionary is constructed in two methods and they are described below.

1. Dictionary based on Random Select: In this method, selection of fingerprint patches is done randomly and these randomly selected patches are arranged as columns of dictionary matrix [11].
2. Dictionary based on K-SVD: K-SVD is a K-means Singular Value Decomposition algorithm. It is a dictionary learning algorithm using K- means clustering method [16].K-SVD algorithm finds sparse

coefficients and updates the dictionary atoms alternatively.

The dictionary is obtained by iteratively solving an optimization problem (6) using Orthogonal Matching Pursuit (OMP) method.

$$\min_{A,X} \|Y - DX\|_F^2, such\ that\ \forall i, \|X_i\|_0 < T \quad \text{..... (6)}$$

where Y represents column of patch, A represents a dictionary, X represents sparse coefficient and T is sparsity constraint.

B. Sparse encoding and updating dictionary

For current dictionary sparse coefficients $x_i$ are calculated by solving equation (8) Orthogonal Matching Pursuit (OMP) method. Then the dictionary column $d_k$ is updated for given $x_i$. Set $\omega_k$ such that it contains non-zero coefficients of $x_i$ and overall error matrix is computed by using equation (9).

$$E_k = Y - \sum_{j \neq k} D_j X_T^j \quad \text{.... (9)}$$

$E_k$ corresponding to $\omega_k$ is chosen to obtain $E_k^{?T}$, then SVD decomposition is applied as $E_k^R = U\Delta V^T$, then the first column of U is chosen as updated dictionary column $d_k$, then updated the coefficient vector $x_k$ as first column of V multiplied by $\Delta(1,1)$. Thus sparse encoding and updating of dictionary are done alternatively.

C. Quantization and entropy encoding

Sparse coefficients are quantized using uniform quantization and these quantized coefficients are encoded using Huffman encoding to get compressed bit stream.

## V. EXPERIMENTS AND RESULTS

This section describes the experiments on different fingerprints. First the databases used for this study has been described. Experimental result for different dictionary methods is given. Next, experimental result for different patch sizes are described. Then comparison between different three algorithms that is SPARSE, SPIHT, WSQ is been described.

A. Databases used

There are 4 groups of fingerprints are used in this experiments namely:

- DATABASE 1: The public fingerprint database FVC2000: DB1(B), DB2(B), DB3(B), and DB4(B) with 10 persons each with 8 samples per person thus total of 320 fingerprints.
- DATABASE 2: the public fingerprint database FVC2002: DB1(B), DB2(B), DB3(B), and DB4(B) with 10 persons each with 8 samples per person thus total of 320 fingerprints.
- DATABASE 3: The public fingerprint database FVC2002: DB1(B), DB2(B), DB3(B), and DB4(B) with 10 persons each with 8 samples per person thus total of 320 fingerprints.
- DATABASE 4: The MSRIT_ec fingerprint database with 100 persons with 10 samples per person thus total of 1000 fingerprints.

B. Experimental result for different dictionary methods

In this section, the effects of different dictionary methods on fingerprint compression is studied. First method is selecting the patches randomly, and arranged them as columns of dictionary and second method to train the dictionary using KSVD method. These methods are tested for DATABASE3 with patch size = 12 × 12. In this experiment PSNR is computed from equation (10) and CR (Compression Ratio) from equation (11).

$$\text{PSNR} = 10 * log_{10} \frac{255\text{^}2}{MSE} \quad \text{..... (10)}$$

Where, MSE is Mean Square Error which defined as

$$\text{MSE} = \frac{1}{M*N} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [X(m,n) - Y(m,n)]^2 \quad \text{..... (11)}$$

Where, X (m, n) represents Original Image and Y (m, n) represents Reconstructed image, M and N indicates number of rows and column in the image respectively.

$$CR = \frac{Number\ of\ bits\ in\ Original\ Image}{Number\ of\ bits\ in\ compressed\ Image} \quad \text{..... (12)}$$

Figure 4 and table 3 represents performance of proposed algorithm under different dictionary methods. Vertical axis indicates average PSNR values for different compression ratios indicated in horizontal axis. Experimental results shows that KSVD method performs better compared Random select method. Therefore KSVD method is used for further experiments.

C. Experimental result for different patchsizes

This section describes experimental results for different patch sizes 8 × 8, 10 × 10, 12 ×12, 16 × 16 are described. Figure 5 shows the dictionary with patch size of 12×12. Figure 6 and table 4 represents average performance of sparse algorithm under different patch sizes for DATABASE1.

| CR | PSNR(dB) | |
|---|---|---|
| | KSVD | Random Select |
| 20 | 33.2 | 31.87 |
| 22.5 | 32.52 | 31.42 |
| 25 | 32.03 | 31.06 |
| 27.5 | 31.62 | 30.91 |
| 30 | 31.27 | 30.58 |
| 32.5 | 30.91 | 30.39 |
| 35 | 30.53 | 30.12 |
| 40 | 30.1 | 28.3 |

Table 3: performance of proposed algorithm for different Dictionary methods for DATABASE3.

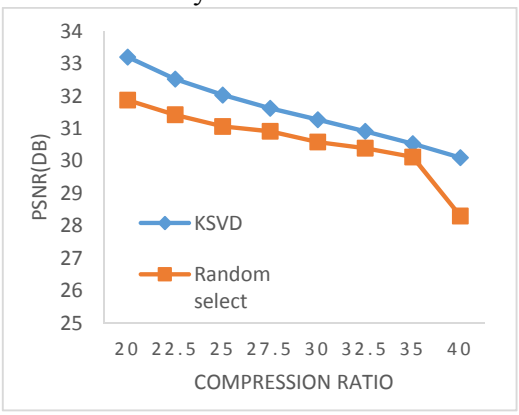


Figure 4. Performance of sparse algorithm under different dictionary method for DATABASE3.

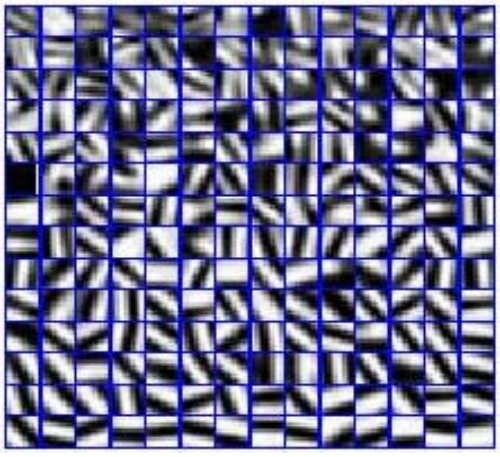

Figure 5 Dictionary with patch size = 12×12

Figure 7 and table 5 represents average performance of sparse algorithm under different patch sizes for DATABASE 2. Figure 8 and table 6 represents average performance of sparse algorithm under different patch sizes for DATABASE 3. Figure 9 and table 7 represents average performance of sparse algorithm under different patch sizes for DATABASE 4. Vertical axis indicates average PSNR values for different compression ratios indicated in horizontal axis. Experimental results shows that 8×8, 10×10 performs better compared to patch sizes of 12×12, 16×16 but it consumes more time for compression and decompression. Therefore patch size of 12×12 is used further in our experiments.

| CR | PSNR(dB) | | | |
|---|---|---|---|---|
| | 8*8 | 10*10 | 12*12 | 16*16 |
| 20 | 32.8 | 32.45 | 32.23 | 31.86 |
| 22.5 | 32.297 | 32.18 | 31.83 | 31.51 |
| 25 | 31.75 | 31.55 | 31.22 | 31.18 |
| 27.5 | 31.15 | 31.08 | 30.97 | 30.76 |
| 30 | 30.68 | 30.65 | 30.41 | 30.17 |
| 32.5 | 30.02 | 30.16 | 30.19 | 30.16 |
| 35 | 29.995 | 30.12 | 30.155 | 30.157 |
| 40 | 28.44 | 28.23 | 28.43 | 28.33 |

Table 4 represents average performance of sparse algorithm under different patch sizes for DATABASE 1.

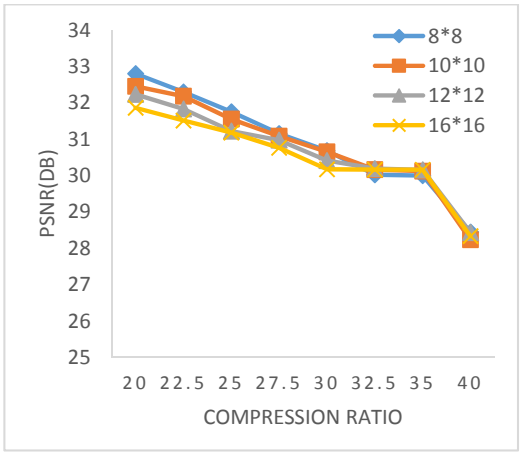


Figure 6 represents performance of sparse algorithm under different patch sizes for DATABASE 1.

| CR | PSNR(dB) | | | |
|---|---|---|---|---|
| | 8*8 | 10*10 | 12*12 | 16*16 |
| 20 | 33.69 | 33.45 | 33.28 | 33.04 |
| 22.5 | 33.26 | 33.13 | 33.02 | 32.83 |

| | | | | |
|---|---|---|---|---|
| 25 | 32.73 | 32.45 | 32.24 | 32.11 |
| 27.5 | 32.32 | 32.09 | 31.23 | 31.23 |
| 30 | 31.68 | 31.55 | 31.14 | 31.14 |
| 32.5 | 31.16 | 31.18 | 31.19 | 31.21 |
| 35 | 30.82 | 31.01 | 31.04 | 31.13 |

Table 5 represents performance of sparse algorithm under different patch sizes for DATABASE 2.

D. Comparison different compression algorithms

This section describes comparison between Sparse, SPIHT, WSQ algorithms.

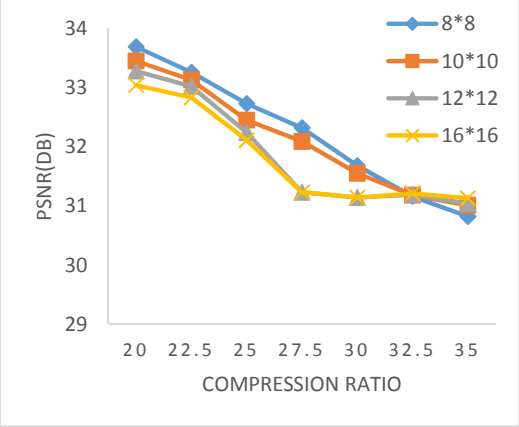


Figure7 represents performance of sparse algorithm under different patch sizes for DATABASE 2.

| CR | PSNR(dB) | | | |
|---|---|---|---|---|
| | 8*8 | 10*10 | 12*12 | 16*16 |
| 20 | 33.94 | 33.35 | 33.14 | 32.98 |
| 22.5 | 33.27 | 33.04 | 32.87 | 32.51 |
| 25 | 32.86 | 32.76 | 32.35 | 32.013 |
| 27.5 | 32.21 | 32.27 | 32.23 | 31.63 |
| 30 | 31.65 | 31.45 | 31.21 | 31.13 |
| 32.5 | 31.16 | 31.15 | 30.87 | 30.81 |
| 35 | 30.88 | 30.92 | 30.94 | 30.96 |

Table 6 represents performance of sparse algorithm under different patch sizes for DATABASE 3.

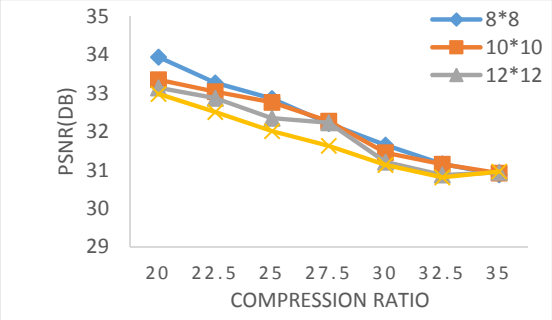


Figure 8 represents average performance of sparse algorithm under different patch sizes for DATABASE 3.

| CR | PSNR(dB) | | | |
|---|---|---|---|---|
| | 8*8 | 10*10 | 12*12 | 16*16 |
| 20 | 32.83 | 32.45 | 32.21 | 31.85 |
| 22.5 | 32.31 | 32.04 | 31.87 | 31.53 |
| 25 | 31.87 | 31.54 | 31.23 | 31.09 |
| 27.5 | 31.35 | 31.13 | 31.09 | 30.98 |
| 30 | 30.54 | 30.22 | 30.13 | 30.06 |
| 32.5 | 30.11 | 30.04 | 29.98 | 29.99 |
| 35 | 29.96 | 29.97 | 29.96 | 29.97 |

Table 7 represents performance of sparse algorithm under different patch sizes for DATABASE 4.

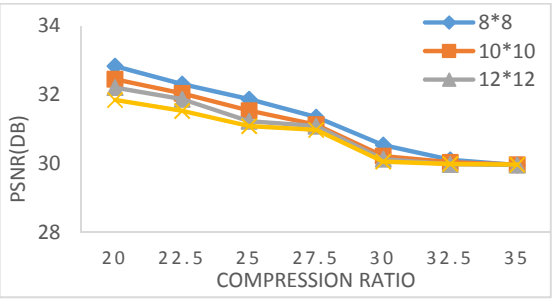


Figure 9 represents performance of sparse algorithm under different patch sizes for DATABASE 4.

| CR | PSNR(dB) | | |
|---|---|---|---|
| | Sparse | WSQ | SPIHT |
| 20 | 32.23 | 31.34 | 32.25 |
| 22.5 | 31.83 | 30.96 | 31.81 |
| 25 | 31.22 | 30.42 | 31.17 |
| 27.5 | 30.97 | 30.05 | 30.72 |
| 30 | 30.41 | 29.99 | 30.23 |
| 32.5 | 30.19 | 29.62 | 29.83 |
| 35 | 30.155 | 29.25 | 29.71 |
| 40 | 28.7 | 27.1 | 27.33 |

Table 8 represents performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 1.

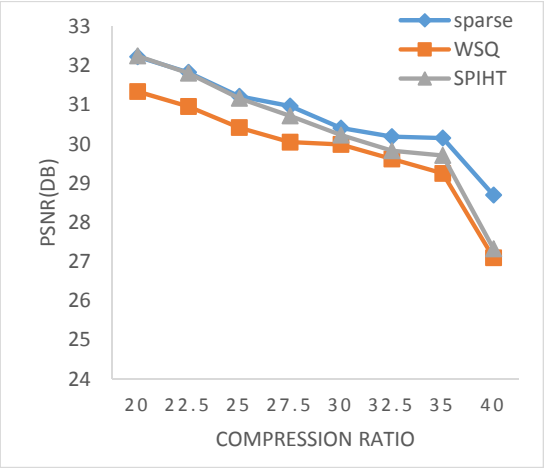


Figure 10 represents performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 1.

| CR | PSNR(dB) | | |
|---|---|---|---|
| | Sparse | WSQ | SPIHT |
| 20 | 33.28 | 32.54 | 33.43 |
| 22.5 | 33.02 | 32.13 | 33.07 |
| 25 | 32.24 | 31.42 | 32.21 |
| 27.5 | 31.23 | 31.07 | 31.17 |
| 30 | 31.14 | 30.31 | 30.92 |
| 32.5 | 31.19 | 29.93 | 30.55 |
| 35 | 31.04 | 29.48 | 30.24 |

Table 9 represents performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 2.

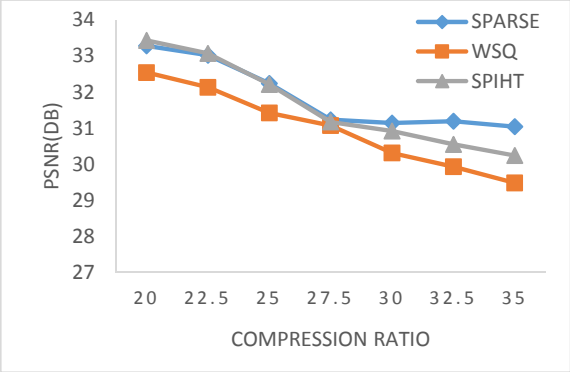


Figure 11 represents performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 2.

| CR | PSNR(dB) | | |
|---|---|---|---|
| | Sparse | WSQ | SPIHT |
| 20 | 33.14 | 33.136 | 33.23 |
| 22.5 | 32.87 | 32.854 | 32.92 |
| 25 | 32.35 | 32.313 | 32.336 |
| 27.5 | 32.23 | 32.217 | 32.227 |
| 30 | 31.21 | 31.026 | 31.13 |
| 32.5 | 30.87 | 30.743 | 30.52 |
| 35 | 30.44 | 30.298 | 30.32 |

Table 10 represents performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 3.

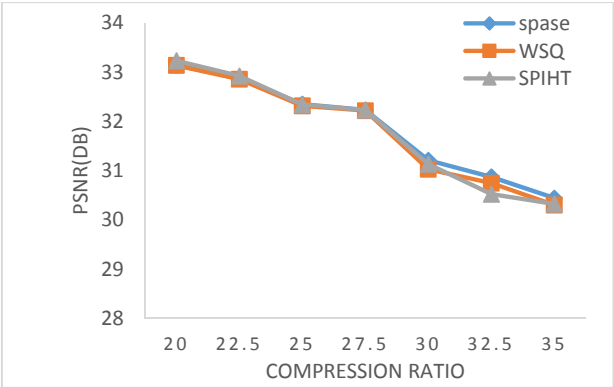


Figure 12 represents performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 3.

Figure 10 and table 8 represents average performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 1. Figure 10 and table 6 represents average performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 2. Figure 11 and table 6 represents average performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 3. Figure 12 and table 7 represents average performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 4.

Vertical axis indicates average PSNR values for different compression ratios indicated in horizontal axis. Experimental results shows that SPARSE algorithm performs better than SPIHT and WSQ in most of the cases. But due to the complexity of SPARSE algorithm, it takes more processing time compared to SPIHT and WSQ. Figure 14, 15, 16 shows sampled result for SPARSE, SPIHT, WSQ algorithms.

| CR | PSNR(dB) | | |
|---|---|---|---|
| | Sparse | WSQ | SPIHT |
| 20 | 31.43 | 30.32 | 31.17 |
| 22.5 | 30.22 | 29.51 | 30.26 |
| 25 | 29.53 | 28.72 | 29.12 |
| 27.5 | 28.21 | 27.17 | 28.44 |

| | | | |
|---|---|---|---|
| 30 | 27.83 | 26.53 | 27.72 |
| 32.5 | 27.04 | 26.03 | 26.92 |
| 35 | 25.65 | 25.42 | 25.63 |

Table 11 represents performance of SPARSE, WSQ, SPIHT algorithms for DATABASE 4.

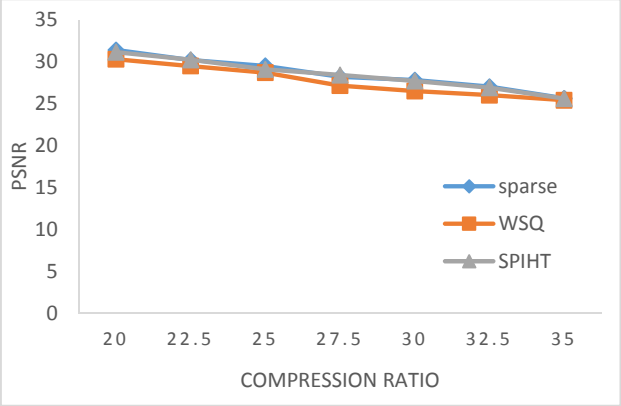


Figure 13: represents performance of SPARSE, WSQ, SPIHT algorithms for DATABASE4.

(a) (b) (c) (d)

Figure 14:(a) Original image 101_1 from FVC2002_DB1B,(b) sparse output with PSNR = 33.95 dB, (c) WSQ output with PSNR = 31.72dB, (d) SPIHT output with PSNR = 32.77 dB at compression ratio 30:1

(a) (b) (c) (d)

Figure 15:(a) Original image 101_2 from FVC2002_DB1B, (b) Sparse output with PSNR = 30.04dB, (c) WSQ output with PSNR = 20.38dB, (d) SPIHT output with PSNR = 29.17dB at compression ratio = 35:1

(a) (b) (c) (d)

Figure 16: (a) Original image 11_10 from DATABASE4, Reconstructed image (b) with sparse, PSNR = 29dB (c) with WSQ, PSNR = 21.32dB (d) with SPIHT, PSNR = 28.96dB.

## VI. CONCLUSION AND FUTURE WORK

Fingerprint compression using SPARSE, SPIHT and WSQ algorithm has been described. Experiments are conducted on four set of databases. Dictionary is constructed using Random select and K-SVD. Experimental results shows that the Dictionary Construction based on K-SVD performs better than Random select method. Experiments are conducted on different patch sizes, patch size of 12 × 12 gives better performance compared to other patch sizes. The proposed algorithm is also compared with SPIHT and WSQ. The Experimental results show that proposed algorithm outperforms SPIHT and WSQ in most of the cases.

As a future work optimization algorithm for solving sparse representation need to be investigated, complexity and processing time of the code is need to be reduced. Improvement quantization and entropy encoding part. Different method for constructing dictionary can used to yield better result.

## REFERENCES


[1] ShohrehKasaei, Mohamed Deriche, BoualernBoash,"A novel Fingerprint compression technique using wavelet packet and Pyramid Lattice Vector Quantization," IEEE transaction on Image Processing vol 11, No. 12, Dec-2002, pp no. 1365-1378.

[2] S. Esakkiroyan, T.Veerakumar, V. SenthilMurugan and R. Sudhakar "Fingerprint compression using counterlet transform and multistage vector quantization," International journal of Computer, Information System and Control Engineering, vol. 1, No.3, 2007.

[3] D. Maltoni, D. Miao, A. K. Jain and S. Prabhakar," Handbook of Fingerprint Recognition, 2nd edition, London, U. K. Spinger-Verlag 2009.

[4] VaniPerumal, JagannathaRamaswamy, " An innovative scheme for effectual Fingerprint compression using Beizer Curve representation," International Journal on Computer Science and Information Security, vol. 6, No. 1, 2009, P. no:149-157.

[5] Guangqishao, Yamping Wu, Yong A, Xialo Liu and TiandeGuo,"Finger print image compression based on sparse representation", IEEE transaction on image processing, page no:489-501, vol. 223, no.2, Feb2014.

[6] W. Pennebaker and J. Mitchell, JPEG—Still Image Compression Standard. New York, NY, USA: Van Nostrand Reinhold, 1993.

[7] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG-2000," in Proc. IEEE Data Compress. Conf., Mar. 2000, pp. 523–541.

[8] A. Said, W. A. Pearlman,"A new fast and efficient Image coded based on SPIHT", IEEE transaction on circuits and systems for video Technologies, vol. 6, pp. 243-250,1996.

[9] Jonathan N. Bradley, Christopher M. Brislawn, Tom hopper, "The FBI wavelet/scalar quantization standard for gray-scale fingerprint image compression," Society of photo-optical instrumentation engineers proceedings, pp293-304, volume 1961, visual information proceeding II, Orlando, Florida, April 1993.

[10 ]B. Karunakumar, K. Sathyaprasad, "Wavelet scalar quantization," International journal of advanced networking and applications, pp 141-146, volume 01, Issue:02, April 2009.

[11] Guangqi Shao, Yamping Wu, Yong A, Xialo Liu and Tiande Guo,"Finger print image compression based on Sparse representation", IEEE Transaction on Image Processing, Page No:489-501, vol. 223, No.2, Feb2014.

[12] J.M. Shapiro, "Embedded Image Coding using Zerotrees of Wavelet Coefficients", IEEE Trans. On Signal Processing, pp. 3445-3462, 1993.

[13] David Solomon, "data compression the complete reference", 4th edition, springer, 2011.

[14] K. Sayood, "Introduction to data compression", third edition, Morgan kaufman publisher, 2006.

[15] Naja M I, Afzal, "Finger print image compression based on sparse representation:

a review" , International journal of computer scince and information technology,vol.6,pp.228-231,2015.

[16] Aharon, M. Elad, and A. Bruckstein," K-SVD, an algorithm for designing overcomplete dictionaries for sparse representation", IEEE transactions on signal processing 2006.