A Lossless Medical Image Compression Framework Using Logic Minimization

Swathi Pai M[1], Jacob Augustine[2], Pamela Vinitha Eric[3]
Research Scholar-School of Computer Science and Engineering, Presidency University Bengaluru, 560119, Karnataka, India [1]
Department of CSE (AIML), PES University, Bengaluru, 560059, Karnataka, India [2]
School of Computer Science and Engineering, Presidency University Bengaluru, 560119, Karnataka, India [3]

***Abstract*—Medical image compression is an active research area owing to the growth of the volume of medical image data in digital form. A method for lossless compression of medical images was proposed using logic minimization. The grayscale medical image is split into bit planes, and each bit plane is divided into blocks of fixed size, e.g., 8×4. The binary bit stream resulting from each of these blocks is treated as the output of a Boolean function, and logic minimization is attempted for a compact form. If this step fails to give a compact representation, the bits are stored as such. In this study, an extendable framework for lossless compression of medical images is presented. The bit plane is adaptively divided using a Quadtree to capture large uniform areas as leaf nodes. The non-uniform blocks at the leaf node are subjected to a logic minimization approach, as in the case of fixed-size blocks in previous related work. To improve the result further, the original image is gray-coded as a pre-processing step. On the bit plane, an XOR operation is done for the current block with the neighboring block for redundancy removal. This framework allows further exploration with the incorporation of other Boolean function representation techniques to enhance the compression.**

***Keywords*—Medical image compression; lossless compression; Quadtree; logic minimization; bit plane encoding; XOR operation; gray coding**

I. Introduction

Medical images such as X-rays [1], CT scans [2], and MRIs [3] contain critical diagnostic information and therefore must be stored and transmitted without any loss of information [4]. As healthcare systems increasingly rely on digital imaging, the volume of medical data has grown manyfold, creating challenges in storage, network bandwidth, and real-time clinical access. This has seen the lossless compression methods gain prominence in this area since they are used to maintain image fidelity as well as to decrease size and thereby enable efficient telemedicine workflows, long-term archival, and quicker retrieval of information in hospital information systems [5]. Close to perfect compression ratios and at the same time containing no loss is an unsolved research problem, particularly when it comes to heterogeneous medical imagery, with different textures and intensities. This study proposes a lossless framework of medical image compression that builds upon Quadtree-based [6] spatial decomposition together with Boolean logic minimization. The gray coding method is used on the input grayscale image in order to increase the inter-pixel correlation across bit planes [7]. A hierarchical Quadtree structure is then used to process each bit plane, starting with a 64×64-block then a 32×32 block, and then further on 16×16, and then finally an 8×8 block, depending on block uniformity. The leaf-level code produced during this decomposition is coded with the ESPRESSO logic minimizer in order to leverage the redundancy between binary patterns [8]. There are also blocks that fail to minimize well; these blocks are considered to be incompressible. In such situations, to further reduce entropy, an XOR-based prediction step is then used, with each incompressible block being XORed with its spatial neighbor to extract further regularity. The blocks that have been XOR-processed are then minimized with ESPRESSO, and blocks that return compression coded in this manner. The blocks that fail to yield compression shall be coded as incompressible. The suggested hybrid loss-less compression scheme, consist of gray-coding, multi-level Quadtree partitioning, logic minimization, and XOR-based refinement, is a customized scheme for medical images [9]. With the rise in the need for efficient handling of medical data in a modern healthcare setting, this stratified structure will strive to achieve maximum compression ratio, as well as pixel-perfect reconstruction to meet the increasing demands. The compression can also be optimized by incorporating any other methods to the incompressible blocks at the leaf level through this framework.

II. Related Work

A Quadtree is a hierarchical representation or structure, which recurrently divides an image into smaller and smaller parts [10, 11, 12]. Each block is checked at every stage to find out if the pixels of every block are of the same binary value. In such a way, the block turns into a leaf, otherwise it is divided into four equal quadrants, as illustrated in Fig. 1.

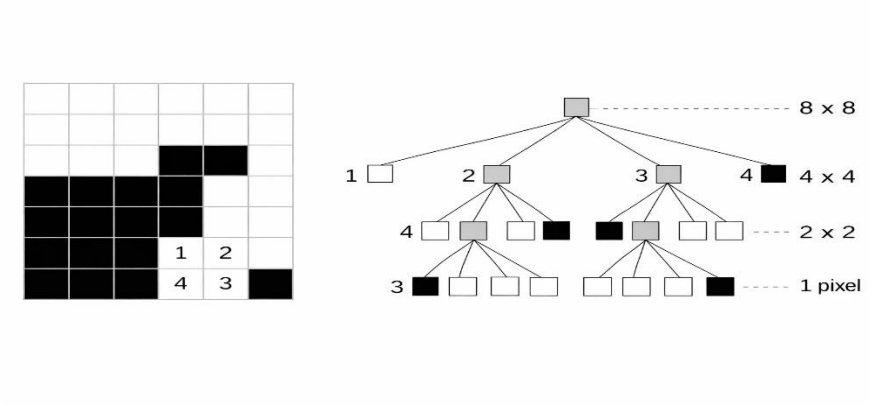

Fig. 1. Quadtree representation.

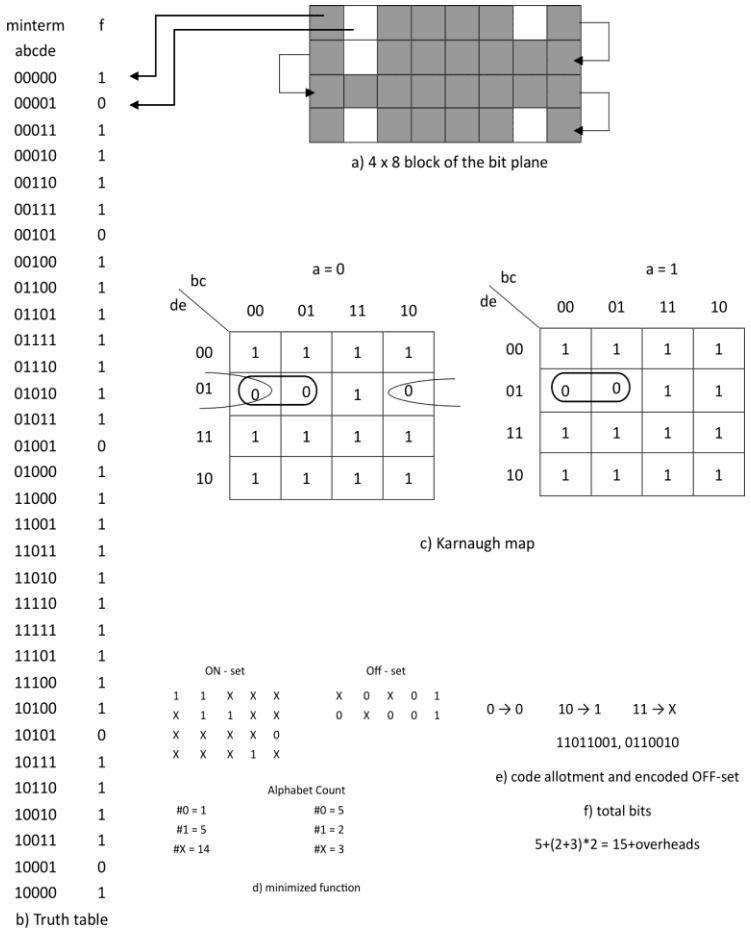

Fig. 2. Function generation and logic coding.

The adaptive representation enables the representation of large homogenous areas by a few nodes, which does not require the storage of redundant pixel data. Thus, for a binary image of size $2^n \times 2^n$, the entire region forms the root of the Quadtree, and subdivisions continue until each leaf contains only 0s or only 1s. Images with large homogeneous areas naturally stop at higher levels of the tree, producing a compact description. In this framework, the grayscale input image (8 bits per pixel) is first separated into eight bit-planes, from the least significant bit s_0 to the most significant bit s_7. Each bit-plane is there- fore a binary image that captures one layer of intensity information.

The Quadtree process is then applied to every bit-plane independently. Initially, each plane is divided into non-overlapping 64 × 64 blocks. This block size is chosen under the assumption that uniform regions are unlikely to extend beyond this scale. Each of these 64 × 64 blocks becomes a Quadtree root and is examined for uniformity. If the block is not homogeneous, it is subdivided to 32 × 32, and subsequently to 16 × 16 and 8 × 8, depending on the spatial variation of pixels [13].

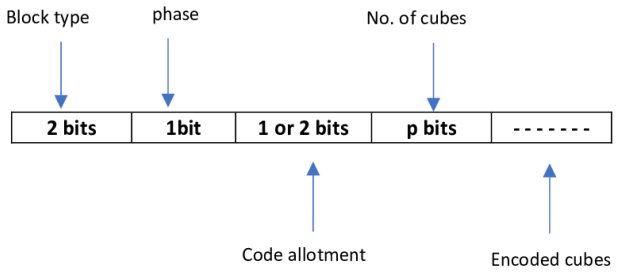

Fig. 3. Format of compressed image.

At the 8×8 stage, a change is incorporated. Rather than the Quadtree process, any non-uniform 8×8 block is further divided vertically to two 8 x 4 blocks. This is refined to capture to vertical structures and narrow column-oriented features that are seen in medical images. In lower bit-planes (s0–s3), these 8 × 4 regions often contain fine-grain details resembling noise, while in the higher bit-planes (s4–s7), they tend to preserve strong anatomical edges [14]. This stage prepares the binary data for logic minimization more effectively. Each 64 × 64 region is tested to ensure the Quadtree-based representation is actually beneficial. If the compressed representation produced by the Quadtree requires more bits than simply storing the block row-wise, the original block is kept instead. A single flag bit is stored to indicate whether Quadtree coding was used:

- 0 – the block is compressed using Quadtree rules
- 1 – the block is stored directly as raw pixels

Two bits are used at each internal node of the Quadtree to indicate the nature of the region:

- 00 – region is entirely black
- 01 – region is entirely white
- 10 – region requires further subdivision
- 11 – leaf node (special handling begins at the 8 × 8 level)

For a mixed region, subdivision continues until the 8 × 8 scale is reached. At this point, the region is split into two 8 × 4 blocks, and each 8 × 4 block is classified using two bits:

- 00 – block is all black
- 01 – block is all white
- 11 – block contains mixed pixels and must be encoded

For mixed 8 × 4 blocks, the corresponding 32-bit pattern is encoded using Boolean logic minimization through the ESPRESSO algorithm. If ESPRESSO produces a representation shorter than the raw block, the minimized form is stored and marked using:

- 0 – logic coding successful

If the minimizer is unable to reduce the block size, raw pixel data is stored instead:

- 1 – logic coding unsuccessful; retain original bits

Such a combination of multi-level Quadtree decomposition, non-uniform 8x8 areas being treated with special care and Boolean minimization allows the representation of structured and unstructured regions in the image to be made efficient and ensure that compression is no higher than the cost of storing the raw block.

A. Logic Coding

Jacob et al. [15] proposed an extension of the block-based binary image coding to the use of logic minimization principles. A grayscale image is split in their method into single bit-planes and each bit-plane is divided into blocks n x m size, both dimensions being powers of two [16]. Each block would belong to one of three groups, namely, all-black, all-white, and interracial. Short predetermined codes are assigned to uniform blocks, whereas mixed blocks get translated into switching functions and simplified using ESPRESSO logic minimizer [17]. The technique makes sure that only minimization causes compression of the number of bits; in case the original block cannot be compressed, the original block is kept [18].

The pixel positions are mapped to minterms by a process known as gray coding, such that adjacent pixels in the image can be translated to adjacent ones in the truth table [19]. This neighboring adds the probability of cube combining in the entity of minimization. In order to maintain this adjacency between rows, the algorithm switches forward and reverse scanning of rows [Fig. 2(a)]. Logic minimization [20] is sometimes capable of encoding a block of 32 pixels with a significantly smaller number of bits than necessary, e.g., 32 bits to around 15 bits, together with any overhead. The conversion of a 4x8 block into a five-variable switching function, the associated Karnaugh map groupings [21], and the number of alphabets in an ON-set and OFF-set cube, respectively, are shown in Fig. 2 [22]. There is always a frequency of cube occurrences and, based on these, prefix codes like 0, 10, and 11 are given to cube types [Fig. 2(e)]. The format of encoding of every block of bit-planes is shown in Fig. 3. A two-bit header identifies the block type:

- 00 for all-black blocks
- 01 for all-white blocks
- 10 for blocks successfully minimized
- 11 for blocks where minimization fails

In the case of incompressible blocks (type 11), original nm bits are simply stored at the end of the header, [23]. Another bit is used to show whether the best ON-set or OFF-set representation has been chosen in compressible blocks. A further code allocation step (summarized in Table I) uses a p-bit field, whose size varies from 1 to 6 depending on the block's minimization characteristics.

TABLE. I. PREFIX CODE ALLOTMENT TO SYMBOLS

Allotment Indicator	Code Allotment		
	0	1	X
0	0	10	11
10	11	0	10
11	10	11	0

The pixel values are replicated through an annual

assessment of the minimized cubes on all minterms of the block during decoding. One of the pixels is given a one in the end of a minterm held by any cube in the ON-set; a zero in the end of a minterm held by the OFF-set, or at the end of a minterm not subsumed by any RAM-ON cube. This subsuming cube- based reconstruction approach guarantees that the representation by the compression creates the binary framework of the block accurately (once the binary structure) [24].

III. Proposed Methodology

The suggested approach (illustrated in Fig. 4) starts with changing the input grayscale picture into its Gray-coded counterpart, such that adjoining intensity levels are a single step apart [25]. The changes make the binary transitions seen within the bit-planes more continuous to allow an even more compressible representation. The source pixel addresses with adjacent pixels represented by one bit difference are mapped to pixel positions using gray coding to allow those pixels to be packed in a narrow band together. The significance of this property is that it conserves the locality of space when a pixel block is viewed as a Boolean function [26]. As an example, a pixel value of 120 (binary 01111000) gets converted to gray code 01000100, the neighboring value 121 (binary 01111001) becomes 01000101, indicating that there is a change in the value of only a single bit between neighboring pixels. The Gray-coded image is then decomposed into eight bit-planes, each containing the 0^{th} to 7^{th} bit of all pixels. Since each bit-plane is now a binary image, further processing is performed independently on every plane. To capture spatial uniformity effectively, each bit-plane is divided into non- overlapping blocks of size 64×64. Every block serves as the root of a quadtree and is recursively split into 32 × 32, 16 × 16, and finally 8 × 8 regions. The splitting of an image recursively is shown in Fig. 5.

A block becomes a leaf whenever its pixels are uniform; otherwise, subdivision continues. This hierarchical partitioning concentrates detailed processing only on non-uniform regions while compactly representing large homogeneous areas. Once the 8 × 8 level is reached, the block is considered the fundamental unit for logic-based encoding.

Each non-uniform block at the finest level is treated as a Boolean switching function whose output corresponds to the binary values of the $n \times m$ pixels in the block [28]. The block is mapped to a function of $\log_2(nm)$ variables, and the widely used ESPRESSO logic minimizer is employed to reduce the number of cubes required to represent this function. Blocks for which minimization yields a reduction in bit count are stored in compressed form, while the remaining ones are labeled as incompressible and preserved in their original pixel format.

To further reduce the number of blocks that remain incompressible after the first round of logic minimization, an XOR-based refinement step shown in Fig. 6 is applied. This step uses a left-predictor model, where each block is predicted using its immediate left neighbor. Let $B(i,j)$ be the current block in row i and column j, and $B(i, j-1)$ be the block to its left. The residual block is computed as:

$$R(i,j) = B(i,j) \oplus B(i,j-1) \quad (1)$$

The first block in every row, $B(i, 0)$, has no left neighbor; therefore, prediction cannot be performed and the block is retained in its original form [29]. When neighboring blocks share similar structures, which is typical in medical images, the XOR operation cancels out the common patterns and produces a residual with a larger number of zeros. For example, consider a left block L and the current block C from a bit plane:

L = 10110110

C = 10100111

The XOR residual R is obtained as

$$R = L \oplus C = 00010001.$$

The resulting residual block contains long runs of zeros, which makes it more suitable for subsequent minimization using the logic-based compression stage. The residual block $R(i, j)$ is therefore passed through ESPRESSO for a second minimization attempt. In the proposed Scheme, heterogeneous leaf blocks are encoded using a compact header structure to indicate the applied transformations. For leaf blocks of size 8 × 8, a one-bit XOR header is first used to signal whether left predictive XOR has been applied between the current block and its immediate left neighbor, and logically minimized.

- 1: indicates that the XOR transformation is performed
- 0: denotes that the original block values are retained

After the XOR operation, the resulting block is subjected to Boolean logic minimization [27] using the ESPRESSO algorithm. After that, a one-bit storage type header is employed to show the state of whether the block is held as raw pixels or as minimized Boolean cubes.

- 1: represents raw storage
- 0: denotes logic-minimized representation

This combined encoding of Gray encoding, adaptive Quadtree partitioning, and separable steps of logic minimization with refinement of XOR is to improve compressibility without any relationships to the image being altered during the encoding process to maintain the lossless input of the reconstructed image.

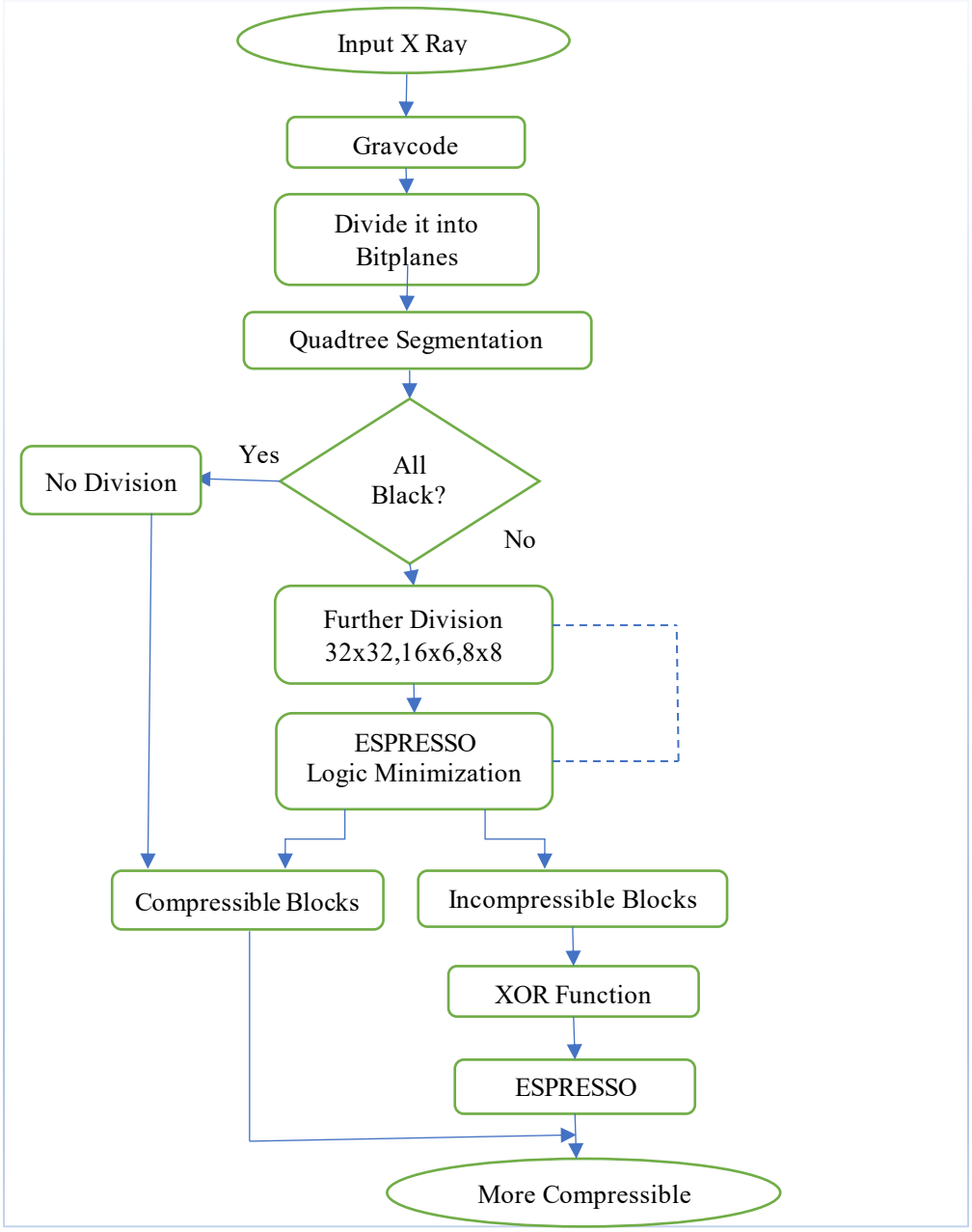

Fig. 4. Flowchart for the related work.

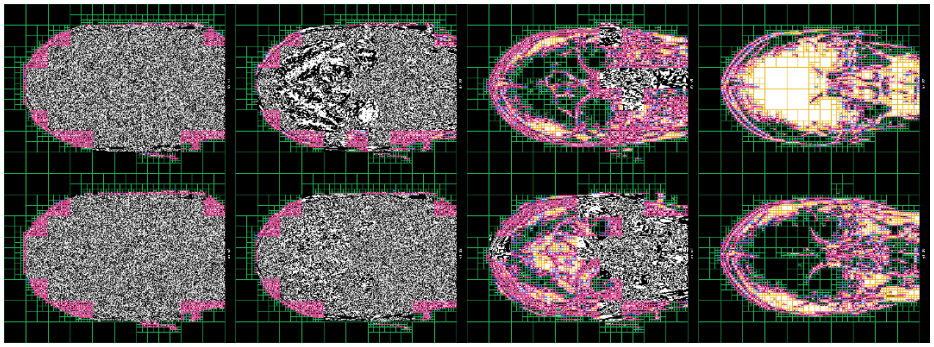

Fig. 5. Recursive Quadtree division.

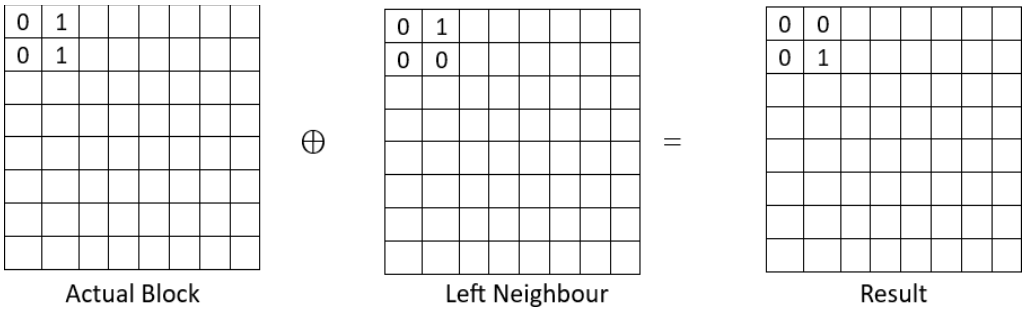

Fig. 6. XOR function.

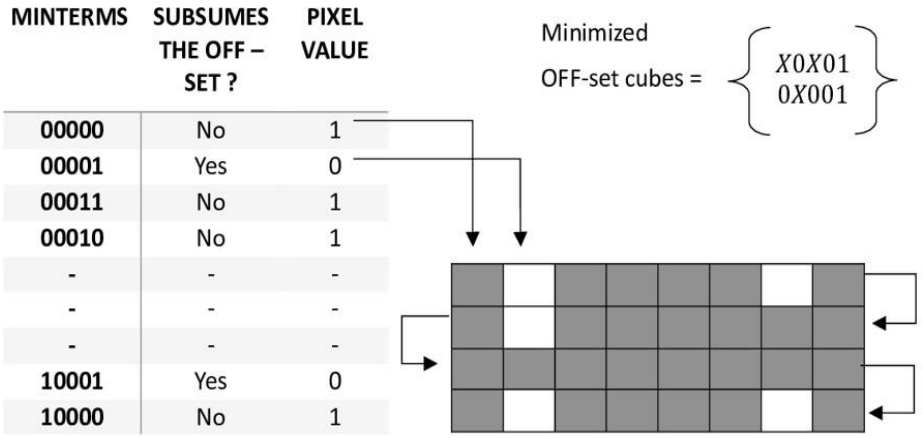

Fig. 7. Decompression scheme.

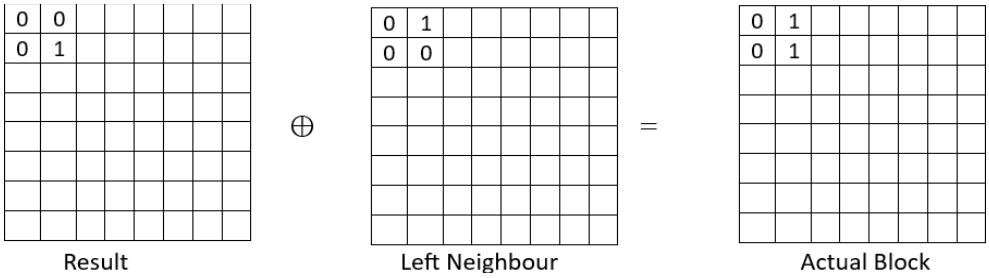

Fig. 8. Reverse XOR function.

A. Decompression Process

During decompression, each bit plane is reconstructed by decoding the compressed blocks in the same sequence used during encoding [29]. For blocks classified as all-black, all-white, or incompressible, recovery is direct since their pixel values were stored without applying logic minimization. For logic-coded blocks, the minimized switching function associated with the block is first rebuilt. The encoded data provide the block type, the phase bit, and the symbol-to-code mapping, from which the minimized ON-set or OFF-set cubes are recovered. Once these cubes are obtained, the switching function is expanded back to its truth-table form using cube subsuming: each minterm generated in Gray-code order is checked against the recovered cubes. A minterm that is contained within any cube of the ON-set is assigned a pixel value of 1, whereas a minterm subsumed by a cube of the OFF-set is assigned 0. Minterms not covered by the minimized cubes receive the complementary value. These values are then placed in their respective positions in the block, producing the reconstructed binary region and is depicted in Fig. 7. For blocks that underwent XOR-based refinement during compression, an additional reversal step is applied, as shown in Fig. 8. Because XOR is self-inverting, the original block C_i is reconstructed by XOR-ing the decoded residual block $R\hat{}i$ with its left predictor Li:

$$Ci = Li \oplus R\hat{}i \qquad (2)$$

Only the first block of each row omits this step, as it was stored directly during compression [30]. By combining logic-coded reconstruction with XOR reversal, each bit plane is recovered exactly as in the original grayscale image.

IV. Results and Discussion

The experimental evaluation was carried out on grayscale X-ray images (shown in Fig. 9) to examine the compression performance of the proposed multi-stage coding scheme which was carried out using Python version *3.13.3* on a Linux platform. Various images which are used here are collected

from Kaggle.com.

The compression efficiency was evaluated using the compression ratio, defined as:

$$Compression\ Ratio = (Size\ of\ original\ image)/(Size\ of\ Compressed\ image) \quad (3)$$

The compression ratios reported in this section, as shown in Table II, reflect the pure logic coding technique [15]. Adaptive segmentation technique [32] and cumulative effect of Gray coding [33] adaptive block partitioning, logic minimization, and XOR-based residual processing is compared with standard PVRG-JPEG Lossless [34] compression technique available.

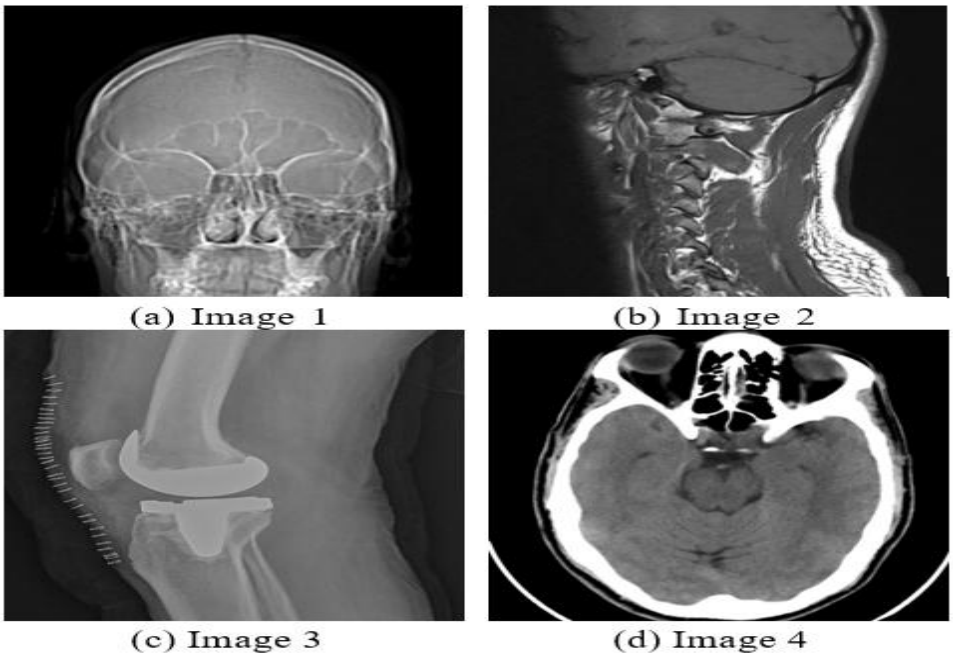

Fig. 9. X-Ray images used for compression.

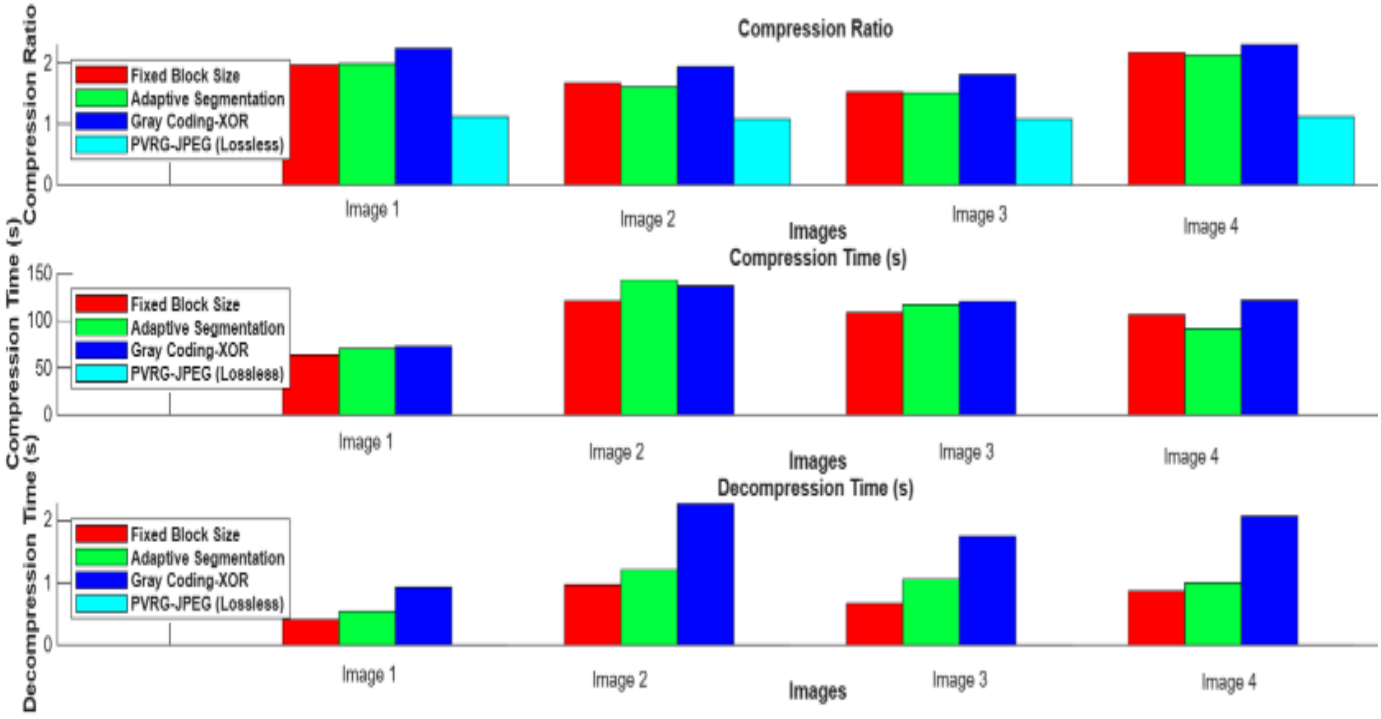

Fig. 10. Comparison of compression efficiency, time, and decompression time for different compression methods.

TABLE. II. SUMMARY OF COMPRESSION RESULTS

Image	Image 1	Image 2	Image 3	Image 4
Logic Coding with Fixed Block Size				
Compression Ratio	1.97	1.67	1.52	2.17
Compression Time (s)	63.8	121.2	108.8	106.5
Decompression Time (s)	0.41	0.97	0.67	0.87
Logic Coding with Adaptive Segmentation				
Compression Ratio	1.99	1.61	1.51	2.13
Compression Time (s)	70.7	143.3	117.3	91.58
Decompression Time (s)	0.54	1.21	1.06	1.00
Gray Coding–XOR Technique				
Compression Ratio	2.24	1.94	1.81	2.30
Compression Time (s)	72.90	137.04	120.60	121.71
Decompression Time (s)	0.93	2.27	1.75	2.07
PVRG-JPEG (Lossless)				
Compression Ratio	1.12	1.08	1.08	1.12
Compression Time (s)	0.02	0.03	0.03	0.03
Decompression Time (s)	0.001	0.001	0.001	0.01

For bit-plane *s*6 of image 1 (as shown in Table III), the Quadtree analysis at the 64 × 64 level shows that 32.50% of blocks are all-black, while no all-white blocks are observed, indicating the presence of large uniform dark regions but absence of bright uniform regions at this scale. The remaining 67.50% of blocks are mixed and therefore require further Quadtree subdivision. At the 32 × 32 level, the percentage of all-black blocks reduces to 18.52%, and all-white blocks increase slightly to 2.56%, while 81.48% of blocks remain mixed, suggesting that significant structural variations still persist. When the block size is reduced to 16 × 16, the proportion of homogeneous blocks in- creases to 16.76% all-black and 2.56% all-white, and the mixed blocks reduce to 80.68%, indicating gradual isolation of uniform regions. At the final 8 × 8 level, 15.67% blocks are all-black and 8.80% are all-white, while a dominant 75.53% blocks become leaf nodes and are forwarded to the logic minimization stage. After applying the ESPRESSO logic minimizer on these 8 × 8 leaf blocks, only 43.63% of blocks are identified as compressible, whereas 56.37% remain incompressible, showing that despite spatial uniformity, logical redundancy is insufficient in many blocks of this higher bit-plane. To further improve compression, left-neighbor predictive XOR coding is applied to these incompressible blocks, followed by a second pass of logic minimization. After XOR, the compressible block percentage becomes 5.53%, while 94.47% blocks still remain incompressible, indicating that although XOR introduces some decorrelation [31] and simplification, the dominant edge and anatomical structures present in the s6 plane limit further Boolean simplification [35]. This behavior is expected for higher bit-planes, where major structural information is preserved, making them less amenable to logic-based compression even after predictive coding. Table III to Table VI shows the statistics of all four images. Fig. 10 compares four approaches to compressing an image: Fixed Block Size, Adaptive Segmentation, Gray Coding-XOR, and PVRG-JPEG (Lossless). The figure is made up of three subplots, which show different performance measurements of each method. The first subplot shows the compression ratio that shows the efficiency of each method in reducing the image size. An increased compression ratio implies increased compression efficiency. The second subplot depicts the compression time which is the duration every method requires to compress the images another important aspect when comparing the speeds of compression techniques. The third subplot revolves around the decompression time as it shows the time needed to restore the images back to their original state. This ratio is most suitable when the application requires details that are real-time. In combination, these subplots present the in-depth evaluation of the compression effectiveness of each approach, processing speed, and decompression effectiveness of each approach, which is of great importance in terms of identifying which one to use in each practice.

TABLE. III. OVERALL STATISTICS FOR IMAGE 1 (PER CENT VALUES)

Block Level	Code Type	s0	s1	s2	s3	s4	s5	s6	s7
	All Black	17.50	17.50	17.50	20.00	25.00	26.25	32.50	37.50
64x64	All White	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.00
	Quadtree	82.50	82.50	82.50	80.00	75.00	73.75	67.50	57.50

	All Black	13.64	13.64	13.64	12.11	10.00	14.83	18.52	7.07
32x32	All White	0.00	0.00	0.00	0.00	0.00	0.50	2.56	6.52
	Quadtree	86.36	86.36	86.36	87.89	90.00	85.17	81.48	86.41
	All Black	7.13	7.35	7.57	7.33	6.94	14.30	16.76	11.79
16x16	All White	0.00	0.00	0.00	0.00	0.12	0.50	2.56	10.22
	Quadtree	92.87	92.65	92.43	92.67	92.94	85.20	80.68	77.99
	All Black	3.63	4.11	4.33	4.62	8.19	13.83	15.67	14.77
8x8	All White Leaf Node	0.00 96.37	0.00 95.89	0.00 95.67	0.27 95.11	3.21 88.61	3.61 82.55	8.80 75.53	14.82 70.41
	Compressible Blocks	0.77	1.51	7.47	16.74	23.21	35.83	43.63	40.87
	Incompressible Blocks	99.23	98.49	92.53	83.26	76.79	64.17	56.37	59.13
Post-XOR Logic Minimization Applied to the above Incompressible Blocks									
Compressible Blocks after XOR		0.03	0.06	0.20	1.04	1.53	2.28	5.53	3.39
Incompressible Blocks after XOR		99.97	99.94	99.80	98.96	98.47	97.72	94.47	96.61

For the compressible blocks, the distribution of ON-set and OFF-set cubes is computed. Between the two, the smaller set is chosen for encoding to ensure a more compact Boolean representation. This adaptive selection contributes to improved compression efficiency without increasing reconstruction complexity. Although in higher bit planes, pixel values mainly represent sharp edges and major structural boundaries. Applying left-predictive XOR across such edges produces irregular residual patterns, which reduces block uniformity and limits the scope for Boolean minimization. As a result, some MSB planes show a slight negative gain in compression after XOR.

TABLE. IV. QUADTREE STATISTICS FOR IMAGE 2 (PER CENT VALUES)

Block Level	Code Type	s0	s1	s2	s3	s4	s5	s6	s7
	All Black	0.00	0.00	0.00	0.00	0.00	26.52	31.06	51.52
64x64	All White	0.00	0.00	0.00	0.00	1.52	0.00	6.06	0.00
	Quadtree	100.0	100.0	100.0	100.0	98.48	73.48	62.88	48.48
	All Black	0.00	0.00	0.00	4.17	0.00	7.99	8.73	26.56
32x32	All White	0.00	0.00	0.00	0.00	6.35	0.26	9.34	0.00
	Quadtree	100.0	100.0	100.0	95.83	93.65	91.75	81.93	73.44
	All Black	0.00	0.00	0.33	7.61	1.28	12.92	6.43	30.32
16x16	All White	0.05	0.57	0.85	1.14	8.83	1.47	16.64	4.52
	Quadtree	99.95	99.43	98.82	91.25	89.89	85.60	76.93	65.16
	All Black	0.14	0.98	5.43	8.34	5.14	12.65	8.81	32.24
8x8	All White Leaf Node	1.08 98.78	2.31 96.71	2.43 92.14	4.62 87.05	12.11 82.75	8.31 79.04	18.76 72.43	11.28 56.48
	Compressible Blocks	9.26	14.55	20.14	21.33	31.99	34.85	43.15	42.91
	Incompressible Blocks	90.74	85.45	79.86	78.67	68.01	65.15	56.85	57.09
Post-XOR Logic Minimization Applied to the above Incompressible Blocks									
Compressible Blocks after XOR		0.23	0.41	0.63	1.35	1.39	2.25	2.64	3.96
Incompressible Blocks after XOR		99.77	99.59	99.37	98.65	98.61	97.75	97.36	96.04

TABLE. V. QUADTREE STATISTICS FOR IMAGE 3 (PER CENT VALUES)

Block Level	Code Type	s0	s1	s2	s3	s4	s5	s6	s7
	All Black	0.00	0.00	0.00	0.00	0.00	0.93	14.81	38.89
64x64	All White	0.00	0.00	0.00	0.00	0.93	3.70	8.33	0.00
	Quadtree	100.0	100.0	100.0	100.0	99.07	95.37	76.85	61.11
	All Black	0.00	0.00	0.00	0.00	0.00	2.91	16.57	28.79
32x32	All White	0.00	0.00	0.00	0.00	1.87	12.14	20.18	7.20
	Quadtree	100.0	100.0	100.0	100.0	98.13	84.95	63.25	64.02

	All Black	0.00	0.00	0.00	0.29	3.33	6.29	14.88	23.22
16x16	All White	0.00	0.00	0.00	0.00	5.48	12.50	19.52	9.76
	Quadtree	100.0	100.0	100.0	99.71	91.19	81.21	65.60	67.01
	All Black	0.00	0.00	0.06	2.18	6.61	12.95	16.61	25.39
8x8	All White Leaf Node	0.00 100.0	0.00 100.0	0.07 99.87	2.23 95.59	10.51 82.88	11.81 75.24	20.05 63.34	12.36 62.25
	Compressible Blocks	0.06	3.33	20.53	38.20	41.99	42.37	38.42	49.02
	Incompressible Blocks	99.94	96.67	79.47	61.80	58.01	57.63	61.58	50.98
Post-XOR Logic Minimization Applied to the above Incompressible Blocks									
Compressible Blocks after XOR		0.00	0.06	0.08	0.48	0.73	1.13	1.55	3.13
Incompressible Blocks after XOR		100.0	99.94	99.92	99.52	99.27	98.87	98.45	96.87

TABLE. VI. QUADTREE STATISTICS FOR IMAGE 4 (PER CENT VALUES)

Block Level	Code Type	s0	s1	s2	s3	s4	s5	s6	s7
	All Black	11.11	12.05	13.19	12.50	12.50	12.50	13.19	28.47
64x64	All White	0.00	0.00	0.00	0.00	0.00	0.00	4.86	0.00
	Quadtree	88.89	87.50	86.81	87.50	87.50	87.50	81.94	71.53
	All Black	9.18	9.13	9.00	9.72	9.92	10.12	11.23	24.76
32x32	All White	0.00	0.20	0.20	0.40	0.40	1.98	13.35	1.70
	Quadtree	90.82	90.67	90.80	89.88	89.68	87.90	75.42	73.54
	All Black	7.20	7.82	7.71	7.89	8.08	9.09	11.52	25.58
16x16	All White	3.33	5.42	5.78	5.68	6.36	13.71	19.66	14.69
	Quadtree	89.46	86.76	86.51	86.42	85.56	77.20	68.82	59.74
	All Black	5.03	6.15	6.52	6.77	9.63	12.61	15.97	30.39
8x8	All White Leaf Node	4.52 90.44	4.98 88.87	5.19 88.29	5.48 87.76	8.14 82.22	18.51 68.88	20.18 63.85	14.12 55.49
	Compressible Blocks	4.73	4.54	13.30	31.12	36.00	40.57	44.27	50.90
	Incompressible Blocks	95.27	95.46	86.70	68.88	64.00	59.43	55.73	49.10
Post-XOR Logic Minimization Applied to the above Incompressible Blocks									
Compressible Blocks after XOR		0.09	0.06	0.42	0.62	1.31	2.07	3.43	3.42
Incompressible Blocks after XOR		99.91	99.94	99.58	99.38	98.69	97.93	96.57	96.58

Overall, the results demonstrate that progressive spatial refinement down to 8×8 blocks, combined with selective XOR-based preprocessing of incompressible regions, enhances the effectiveness of Boolean function minimization across all bit planes.

V. CONCLUSION AND FUTURE SCOPE

A lossless compression framework for grayscale medical images using logic minimization is proposed. Core of the scheme is treating the bit stream at the blocks of the bit planes of the image as a Boolean function, and representing it in its minimized form. All blocks do not yield a smaller representation this way. When a block does not result in a smaller representation, the bits are stored as such. A switching function of n variables has 2^n minterms which are combined by the logic minimizer ESPRESSO into cubes. Each minterm is a vertex on a binary hypercube and the minimizer try to cover the vertices belonging to ON set into a cube which is a compact representation. ESPRESSO minimizer was developed for VLSI synthesis (of Programmable Logic Arrays) of Boolean functions with many variables. This minimizer is not the optimal one in the compression experiment where the number of variables is 5 or 6 and not 50. This minimizer was used owing to the fact that it is a proven tool and good enough to establish the proof of concept though not optimal in terms of execution time. It is always possible to create an optimal minimizer suitable for functions with a smaller number of variables. So, the time of execution reported for both compression and decompression can easily be improved with an optimized logic minimizer.

The total combination of functions for an n variable Boolean function is $2^{\{2^n\}}$. One of the functions is tautology (always 1) and the other is XOR function. Cube based minimizer do not yield optimal representation for all type of functions. For example, XOR function has a poor representation in the cube form but has an efficient representation if Binary Decision Diagrams (BDDs) are used. It is planned to explore these techniques to target blocks which are incompressible in the current scheme. As there is a general framework, new approaches can be incorporated easily.

The experiment based on the method proposed in this study is an improvement on the coding using fixed block size on the bit planes. With Quadtree, gray coding and prediction, an improvement in compression ratio is achieved though not significant over the previous method based on fixed block size. Incorporation of other techniques to handle incompressible blocks can be tried to push up the compression ratio. By looking at the statistics presented in experimental result, it is evident that some of the incompressible blocks will yield result with another representation of Boolean functions. These are some of the future directions planned for exploration.

REFERENCES

[1] Agnes J, Gunasundari S. Prediction of Lung Disease using Deep Learn- ing Techniques in Chest X-Ray Images: A Systematic Review. In2025 5th International Conference on Soft Computing for Security Applica- tions (ICSCSA) 2025 Aug 4 (pp. 1406-1416). IEEE.

[2] Shatnawi MQ, Abuein Q, Al-Quraan R. Deep learning-based approach to diagnose lung cancer using CT-scan images. Intelligence-Based Medicine. 2025 Jan 1;11:100188.

[3] Bonato B, Nanni L, Bertoldo A. Advancing precision: A comprehensive review of MRI segmentation datasets from brats challenges (2012–2025). Sensors (Basel, Switzerland). 2025 Mar 15;25(6):1838.

[4] Sherly Kanaga Priya P, Helen Sulochana C. Learned Lossless Image Compression Based on Optimal Kernel Transformer Approach. The Eu- ropean Journal on Artificial Intelligence. 2025 May;38(2):198-216.

[5] Rath RK, Hazra A, Nanda R. A Lossless Healthcare Data Compression Approach Using Near-Edge Computing. InIndustry 5.0: Key Tech- nologies and Drivers 2025 Apr 4 (pp. 69-92). Cham: Springer Nature Switzerland.

[6] Mahdi SH, Sahy SA, Hasoon JN. Video Compression Using Quadtree Decomposition and Bitplane Coding. JOIV: International Journal on Informatics Visualization. 2025 Nov 30;9(6).

[7] Dvoinishnikov SV, Zuev VO, Bakakin GV, Pavlov VA. Gray code image processing algorithm for measuring the three-dimensional geometry of objects with complex profiles. Measurement Techniques. 2025 Apr 1:1-8.

[8] Gurunath B. Logic minimization algorithms for VLSI applications (Doc- toral dissertation).

[9] Sato Y, Tezuka H, Kondo R, Yamamoto N. Quantum algorithm for partial differential equations of nonconservative systems with spatially varying parameters. Physical Review Applied. 2025 Jan 1;23(1):014063.

[10] Hanan Samet. The quadtree and related hierarchical data structures. ACM Computing Surveys (CSUR), 16(2):187–260, 1984.

[11] Hanan Samet. An overview of quadtrees, octrees, and related hierarchi- cal data structures. Theoretical Foundations of Computer Graphics and CAD, pages 51–68, 1988.

[12] Prashant Shukla, Abhishek Verma, Abhishek, Shekhar Verma, and Man- ish Kumar. Interpreting svm for medical images using quadtree. Multi- media Tools and Applications, 79(39):29353–29373, 2020.

[13] Swathi Pai M, Shyam P Joy, and Jacob Augustine. Lossless compression of grayscale images via representation of bit planes as minimized boolean functions. International Journal of Electronics and Electrical Engineering (IJEEE), 2025.

[14] Mahdi SH, Sahy SA, Hasoon JN. Video Compression Using Quadtree Decomposition and Bitplane Coding. JOIV: International Journal on Informatics Visualization. 2025 Nov 30;9(6).

[15] Chaudhay AK, Augustine J, Jacob J. Lossless compression of images using logic minimization. InProceedings of 3rd IEEE International Con- ference on Image Processing 1996 Sep 19 (Vol. 1, pp. 77-80). IEEE.

[16] Augustine J, Feng W, Jacob J. Logic minimization based approach for compressing image data. InProceedings of the 8th International Confer- ence on VLSI Design 1995 Jan 4 (pp. 225-228). IEEE.

[17] Choi J, Lee S, Kang K, Suh H. Lightweight Machine Learning Method for Real-Time Espresso Analysis. Electronics. 2024 Feb 19;13(4):800.

[18] Butler JT, Sasao T. Boolean functions for cryptography. InProgress in Applications of Boolean Functions 2022 Mar 17 (pp. 33-53). Cham: Springer International Publishing.

[19] PejaŚ J, Cierocki Ł. Reversible data hiding scheme for images using gray code pixel value optimization. Procedia Computer Science. 2021 Jan 1;192:328-37.

[20] Nazemi M, Kanakia H, Pedram M. Heuristics for million-scale two-level logic minimization. In2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD) 2021 Nov 1 (pp. 1-7). IEEE.

[21] Wang C, Tao Y. Karnaugh maps of logical systems and applications in digital circuit design. Circuits, Systems, and Signal Processing. 2020 May;39(5):2245-71.

[22] Van Aerschot W, Jansen M, Bultheel A. Normal offsets for digital image compression. InWavelet Applications in Industrial Processing III 2005 Nov 7 (Vol. 6001, pp. 148-159). SPIE.

[23] Liu X, Abd-Elmoniem KZ, Stone M, Murano EZ, Zhuo J, Gullapalli RP, Prince JL. Incompressible deformation estimation algorithm (IDEA) from tagged MR images. IEEE transactions on medical imaging. 2011 Sep 19;31(2):326-40.

[24] Swathi PM, Nandan RP, Augustine J. Logic minimisation and compres- sion—A lossless approach for medical imaging. InEmerging Technologies in AI, Computation, Communication, and Cybersecurity (pp. 355-361). CRC Press.

[25] Zhou RG, Sun YJ, Fan P. Quantum image Gray-code and bit-plane scrambling. Quantum Information Processing. 2015 May;14(5):1717-34.

[26] He D, Yang Z, Peng W, Ma R, Qin H, Wang Y. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adap- tive coding. InProceedings of the IEEE/CVF conference on computer vision and pattern recognition 2022 (pp. 5718-5727).

[27] Faroß N, Schwarz S. Gröbner bases for Boolean function minimization. Mathematics in Computer Science. 2025 Dec;19(1):7.

[28] O'Donnell R. Analysis of boolean functions. Cambridge University Press; 2014 Jun 5.

[29] Spratling MW. A review of predictive coding algorithms. Brain and cognition. 2017 Mar 1;112:92-7.

[30] Eldawy A, Alarabi L, Mokbel MF. Spatial partitioning techniques in SpatialHadoop. Proceedings of the VLDB Endowment. 2015 Aug 1;8(12):1602-5.

[31] Descloux A, Grußmayer KS, Radenovic A. Parameter-free image resolu- tion estimation based on decorrelation analysis. Nature methods. 2019 Sep;16(9):918-24.

[32] Anitha V, Murugavalli SJ. Brain tumour classification using two-tier classifier with adaptive segmentation technique. IET computer vision. 2016 Feb;10(1):9-17.

[33] Dvoinishnikov SV, Zuev VO, Bakakin GV, Pavlov VA. Gray code image processing algorithm for measuring the three-dimensional geometry of objects with complex profiles. Measurement Techniques. 2025 Apr 1:1-8.

[34] Clunie DA. Lossless compression of grayscale medical images: effectiveness of traditional and state-of-the-art approaches. Medical Imaging 2000: PACS Design and Evaluation: Engineering and Clinical Issues. 2000 May 18;3980:74-84.

[35] Falkowski BJ. Lossless binary image compression using logic functions and spectra. Computers and Electrical Engineering. 2004 Jan 1;30(1):17- 43.