# Improved Distance Coding of Binary Images by Run Length Coding of the Most Probable Interval

Amir L. Liaghati
The Boeing Company
1100 Redstone Gateway SW, Huntsville, AL

W. David Pan
Dept. of ECE, University of Alabama in Huntsville
Huntsville, AL 35899

***Abstract*—We proposed a new method to improve our previous work on efficient distance-coding of binary images, where we compressed a binary image by applying the bzip2 lossless data compressor on a sequence of intervals, which represent the distances between identical source symbols – either zeros or ones for binary images. Motivated by the observation that a majority of intervals tends to be one, we propose to run-length code this most probable interval independently from the rest of the intervals. Separate Huffman coding tables were used to code the run-lengths of the most probable interval versus other intervals. Consequently, this hybrid coding scheme allows a fraction of one bit to be assigned to the most probable interval on average, as opposed to at least one bit per interval without run-length coding, thereby contributing to about 17% improvement on the compression ratios on some test images.**

***Keywords*—lossless compression, binary images, distance coding, entropy, run-length encoding, Huffman coding, bzip2***

## I. Introduction

Over the past decade, the need for image compression has evolved exponentially. In order to reduce the data transmission bandwidth as well as storage needs, more efficient compression techniques are needed. Images generated in the fields of medicine, aerospace, and defense tend to be extremely large and need to be compressed more efficiently. Depending on the application, lossy and lossless compression can be applied. This research aimed at improving the lossless compression of binary images.

Binary images, also known as bi-level images, are images with pixel values of either 1's or 0's. Although high resolution grayscale images might be useful for human eyes, binary images might be more suitable for some applications, where color or grayscale images may be converted to binary images. Examples include visual sensor networks [1], object detection [2], edge detection [3], morphological operation [4], Region of Interest (ROI) segmentation [5] among many others.

Existing methods for compression of binary images include Huffman coding [6], [7], run-length coding [8], [9], arithmetic coding [10], and geometric-based coding [11], [12], etc. In addition, International standards for binary image compression have been developed, such as JBIG [13], JBIG2 [14] and JPEG 2000 [15]. Also, distance coding methods [16]–[19] have been proposed to compress binary images efficiently.

## II. DISTANCE CODING OF BINARY IMAGES

In our previous work on distance-coding of binary images [20], we traversed the binary image to be compressed and calculate the distances of identical symbols - either zeros or ones in the case of binary images. The amount of compression that can be achieved on the original image depends heavily on the sequence of the intervals generated. Specifically, we introduced a block-based interval-generation scheme, where we first partitioned the image into blocks. We then scanned through each block in both horizontal and vertical directions. The final choice of scan direction for each block was chosen adaptively according to a criterion based on entropy. Entropy is calculated using the following formula

$$H(R) = -\sum_{k=1}^{L} P(r_k)\log P(r_k), \qquad (1)$$

where $L$ is the number of distinct symbols $r_k$ (with $1 \leq k \leq L$) in the interval sequence, and $P(r_k)$ is the probability of each symbol $r_k$ occurring in the sequence. In theory, the smaller the entropy $H(R)$, the higher the compression one could achieve on the sequence. The procedure is summarized in Fig. 1, which shows that the sequence of intervals and direction coding bit for each block are combined in a single data file and compressed by a lossless data compressor such as *bzip2*.

## III. DISTRIBUTION OF SEQUENCE OF INTEVALS

An 8-bit grayscale image of an asteroid is used as an illustrating example (Fig. 2 left). Thresholding operation was applied to the grayscale image to obtain a binary image based on the threshold $T$:

$$I(i,j) = \begin{cases} 1, & \text{if } I(i,j) \geq T. \\ 0, & \text{otherwise.} \end{cases} \qquad (2)$$

where $I(i,j)$ is the pixel values at point $(i,j)$. Next, the block-based adaptive interval-generation method was applied to obtain the sequence of intervals and the direction bits. The histogram of the interval sequence is shown in Fig. 3. In most of the binary images we studied, the interval value of '1' occurs more than 90% of the time, whereas intervals greater than one are much less likely. This is to be expected since either black or white pixels tend to cluster in binary images. Since 90% of the intervals have values being '1', the entropy of the sequence tends to be very small, thereby allowing for efficient compression of the original binary image. If we apply Huffman coding directly on the interval sequence, the most probable interval '1' will be assigned one bit per interval. In the case of binary asteroid image, a total of 60,386 interval values equals to '1', accounting for about 91.43% of the entries in the interval sequence. Therefore, applying the Huffman code

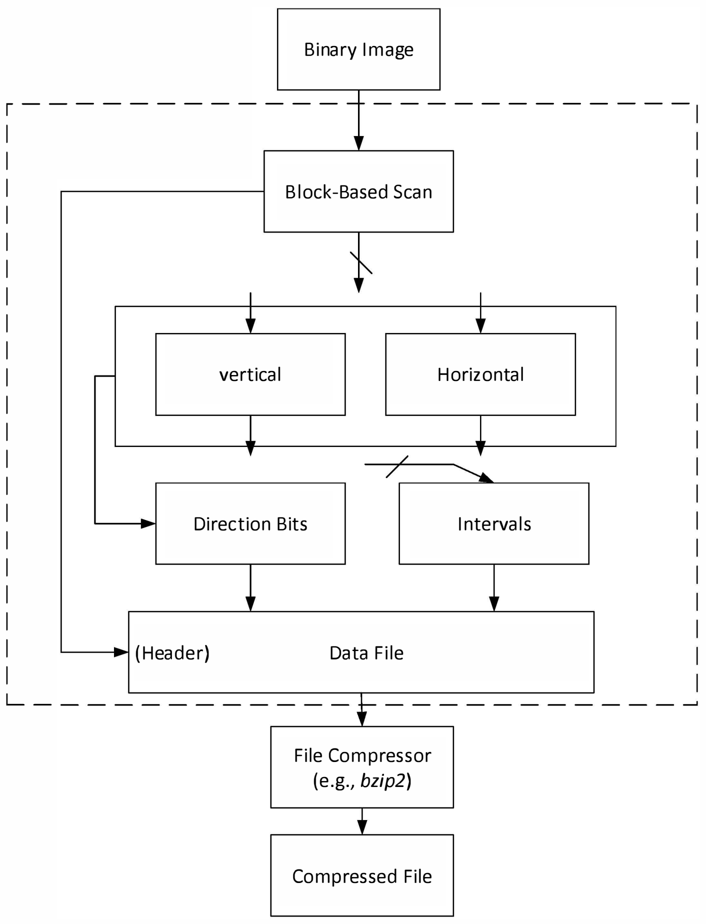

Fig. 1. The encoder of the block-based adaptive interval-generation method.

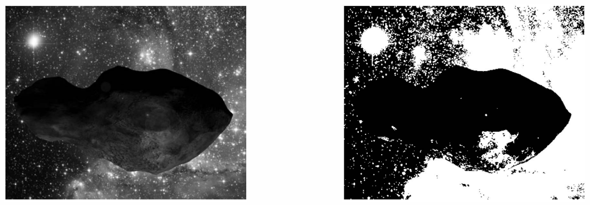

Fig. 2. (Left) Grayscale image of an asteroid; (Right) Binary image obtained after thresholding.

directly on this sequence will generate at least 60,386 bits for those most probable intervals with value being '1', which is a source of inefficiency due to out failure to capture the correlations between these identical entries in the sequence.

## IV. THE PROPOSED METHOD

The idea is to treat the intervals of '1's separately from the rest of the intervals in the sequence with values greater than 1. More specifically, we count the run-lengths of those most probable intervals (with value being '1') and apply the Huffman code on the run-lengths. For all other intervals with values greater than one, we concatenate these intervals and the pointers to the most probable intervals to form a modified sequence of intervals, before applying a separate Huffman code

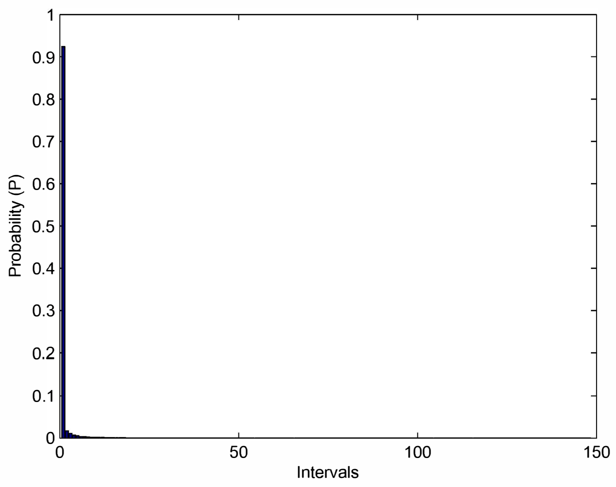

Fig. 3. Distribution of the interval values.

to this modified sequence. Fig. 4 shows the block diagram for the proposed method.

### A. Run-length of intervals with value being 1

The procedure for calculating the run-lengths of '1's in the sequence of intervals is shown in Fig. 5. It starts by checking whether the current interval is '1', followed by a check on the next interval. If the next interval is also a '1', the counter variable "count' will increase by 1. If this is the last interval in the sequence, then the count will be stored in an array called "Count" where all the run-lengths will be stored and that is the end of the loop. If the next interval is not a '1', then the count will be stored in the array "Count", and the procedure repeats. If the current interval is not equal to '1', then we will check for the next interval. And if the next interval is the last interval, we will check whether it is a '1'. If it is a '1', a value '1' will be stored in the current array Count. The case where the last interval is a '1' is a special case. If the interval is not the last interval, the process is repeated until the last interval in the sequence is reached. Also, if the next interval is the last and is not a '1', the procedure will be terminated without counting any '1's. The array "Count" now contains all the run-lengths of 1's.

An example distribution of the run-lengths of intervals with values being '1's is shown in Fig. 6. We will code the run-lengths with Huffman codes. Since the distribution of the run-lengths depends on the source statistics of the original binary image, other compression methods such as Golomb codes can be considered for the coding of run-lengths as well. In this preliminary study, we trained the Huffman code by using the sequence of run-lengths derived from a single image (Fig. 2). The coding table thus obtained would be optimum for this particular sequence only. In practice, we can train the Huffman code based on a set of sequences of run-lengths derived from a large set of test images to achieve some robustness of the coding tables.

### B. Huffman coding for the run-lengths

Fig. 7 shows the Huffman codeword lengths for some run-lengths. It can be seen that as run-lengths exceed 6,

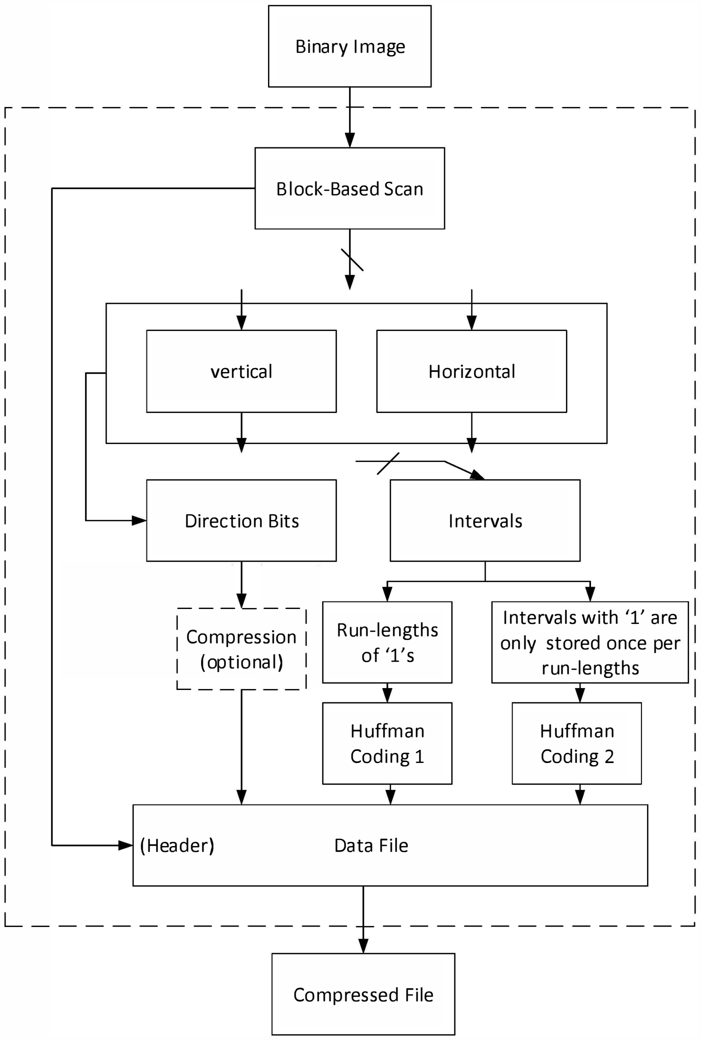

Fig. 4. The proposed encoder.

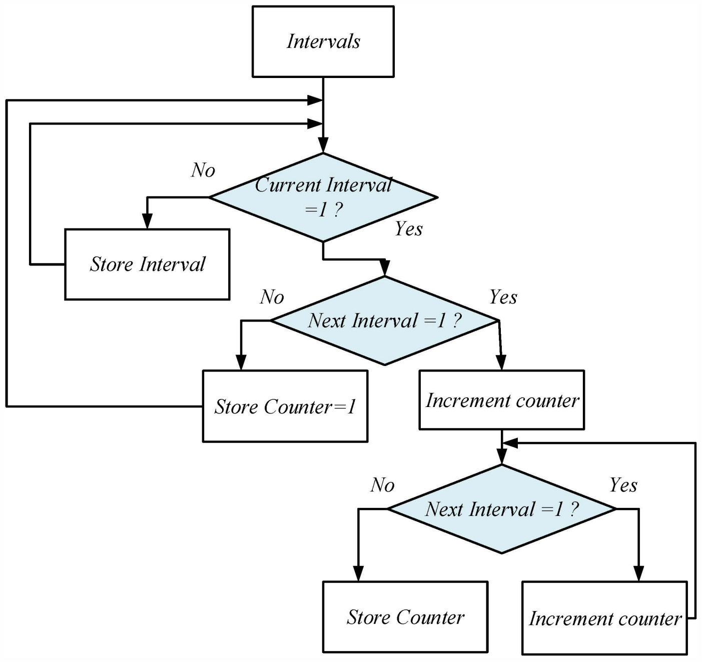

Fig. 5. Block diagram for calculating the run-lengths of interval with value being 1.

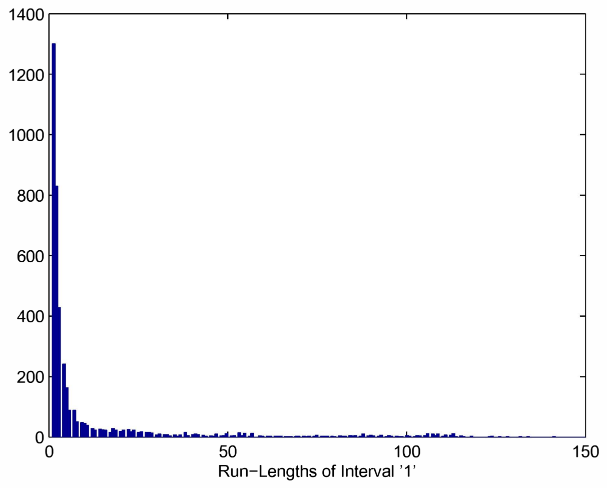

Fig. 6. The histogram of the run-lengths.

the codeword lengths become smaller than the run-lengths. This means run-length coding allow for compression gains on the most probable interval whenever there is more than 6 consecutive intervals with value being 1. For example, when run-length is 48, 10 [1] bits are required to code 48 consecutive intervals with value being 1, which requires at least 48 bits to code if applying Huffman coding directly on the sequence of intervals without using run-length coding. Fig. 8 shows the lengths of the codewords used to code the largest 50 run-lengths for the example image, where a large run-length of 2,989 requires only 12 bits to code, representing a huge saving.

## *C. Generation of the modified sequence of interval values*

For those intervals with value great than 1, we combine them into a modified sequence of intervals, among which one flag bit is interspersed. Fig. 9 shows the block diagram for generating the modified sequence of interval values. For example, given an original sequence of interval values being [2, 1, 1, 1, 1, 5], the modified sequence is [2, 1, 5], where the '1' in the middle is indeed a pointer to an entry in the sequence of run-lengths as described previously in subsection A. The entry being pointed to has a value of 4, meaning four consecutive intervals with identical value being 1 in the original sequence of interval values. This modified sequence of hybrid interval values can be unambiguously reconstructed by the decoder, since any entry equal to 1 means it is a flag bit, which is to be replaced by the actual number of '1' intervals as indicated by the corresponding run-lengths, whereas an entry greater than 1 means it is already an interval value.

[1]Note: In actual implementation, a total of 11 bits would be required, i.e., 10 bits of Huffman codeword, plus 1 flag bit that serves as the pointer to the run-lengths in the modified sequence of block symbols.

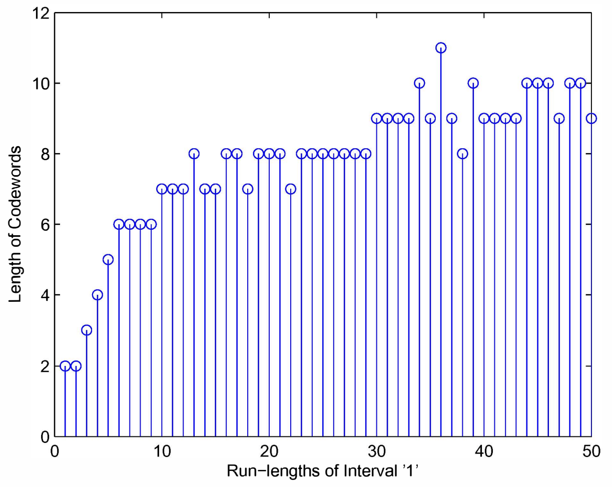

Fig. 7. Huffman codeword lengths for the run-lengths of the most probable interval value. While only the 50 smallest possible run-lengths were shown, the maximum codeword length is 12 bits for all run-lengths.

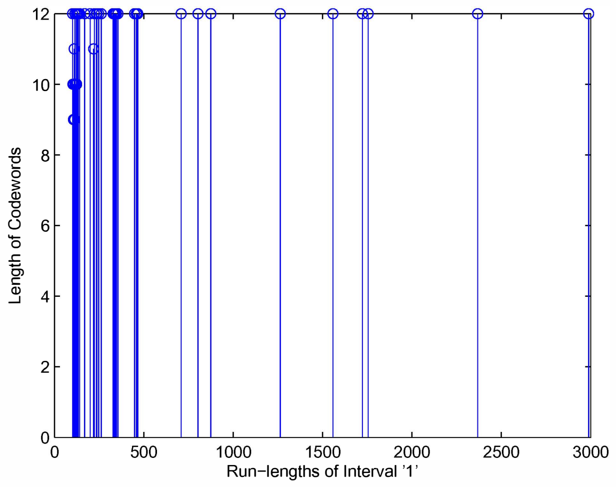

Fig. 8. Length of Huffman codewords used for the largest 50 run-lengths.

## D. Huffman codes for the modified sequence of interval values

We also Huffman coded the modified sequence of interval values to achieve data compression, with some example codeword lengths shown in Fig. 10, where the interval value being 1 means it is a flag bit, which requires exactly a one-bit codeword.

## E. Header information and direction bits

The compressed image file should have a header, which contains the block sizes used in the block-based interval-generation distance coding method, as well as overhead bits used to keep track of the optimal scanning direction chosen for each block can. As an option, the header can be compressed and decompressed losslessly. However, it is advisable to compress the direction bits, which increase significantly as block sizes become smaller.

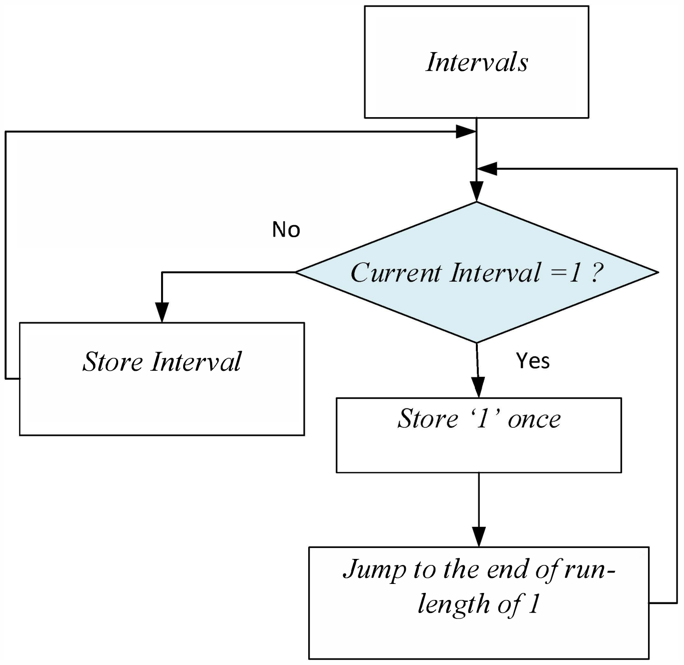

Fig. 9. Diagram for modified sequence of intervals generation.

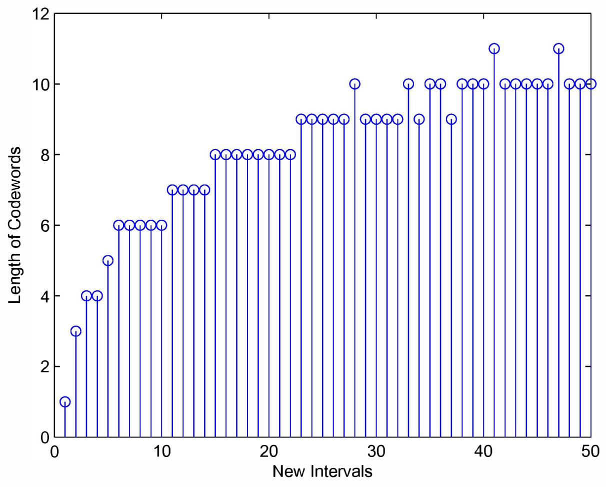

Fig. 10. Lengths of the codewords for the 50 smallest possible hybrid interval values.

## F. Decoder

As shown in Fig. 11, the decoder proceeds by scanning through the modified sequence of hybrid interval values, including values greater than 1 and the flag bits. If a flag bit of '1' encountered, the decoder will refer to the corresponding decoded "Count" array which contains the run-lengths of '1' intervals, and expand the flag bit into the actual number of interval values as determined by the corresponding run-lengths. On the other hand, if values greater than 1 are encountered, the decoder knows that these values are original interval values and thus no expansion operation is needed. As a result, the decoder can reconstruct the original sequence of interval values, based on which the entire image can be reconstructed block by block.

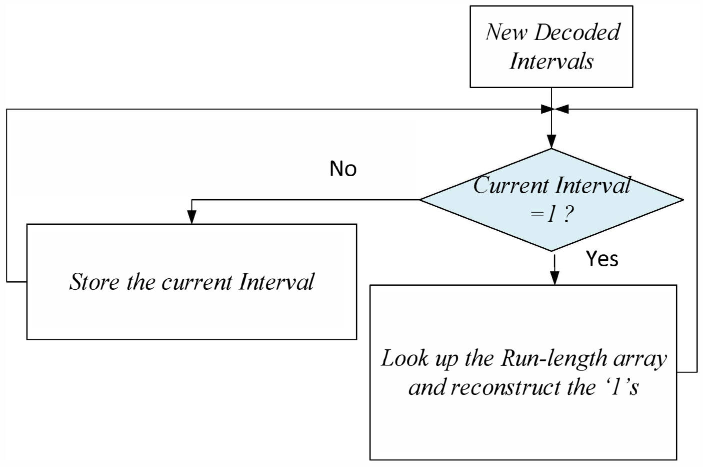
New Decoded Intervals
Current Interval =1 ?
No
Yes
Store the current Interval
Look up the Run-length array and reconstruct the '1's

Fig. 11. Diagram for reconstructing the original sequence of intervals.

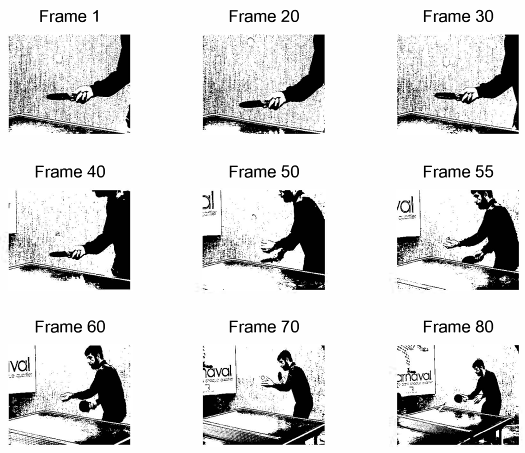

Fig. 12. Selected images from a set of 100 binary images adapted from the "Table Tennis" sequence.

## G. Lossless check

Lossless check was performed at the end of the decoder to make sure the reconstructed binary image is exactly the same as the original image pixel by pixel.

## V. SIMULATION RESULTS

To test the proposed method, a set of 100 binary images were obtained by converting grayscale images to binary images using thresholding from the first 100 frames of the video sequence called "Table Tennis" [21]. Fig. 12 shows 9 selected frames for the purpose of demonstration. As shown in Fig. 13 and Fig. 14, the proposed hybrid coding scheme (the "New Method") improves the compression of the block-based method without run-length coding in [20] (called the "Old Method") by about 17%. Similar improvements were also obtained for some other binary images used in the test, thereby demonstrating the advantage of coding the most probable interval separately using run-length coding. We have also applied the new method on different types of images and compared its performance against the JBIG2 standard. As an

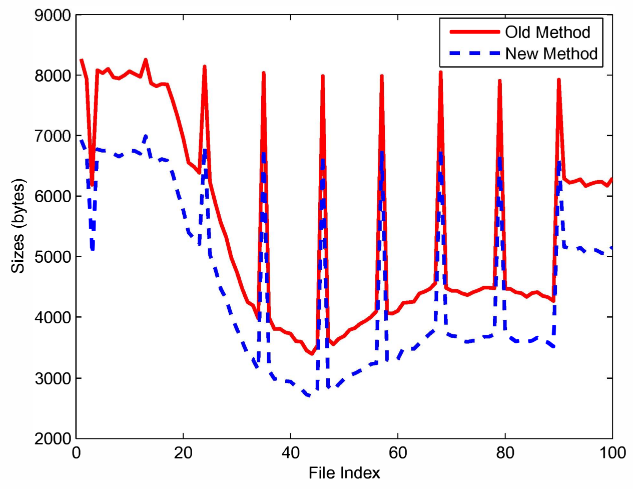

Fig. 13. Compressed file sizes.

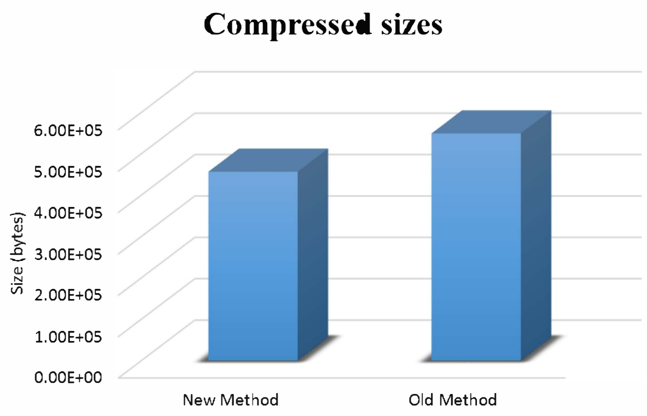

Fig. 14. The overall compressed file sizes for 100 binary images.

example, we try to compress binary image used in soccer player detection [22]. Fig. 15 (left) shows a video frame from the World Cup 2014 final game between Germany and Argentina [23]. Next, we used a basic color detection method to detect the players in blue inside the field. Then the image was converted to a binary image, where only players in blue are shown in black as foreground, the rest of the image was converted to white as background as shown in Fig. 15 (right). We then applied both our method and the JBIG2 standard codec on the image. Four different block sizes were used to compress the image using our proposed method, and every

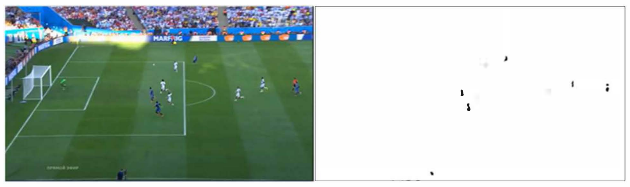

Fig. 15. A video frame (720 × 1272) selected from the scene of the Final Game of World Cup 2014 (left); the binary version (right).

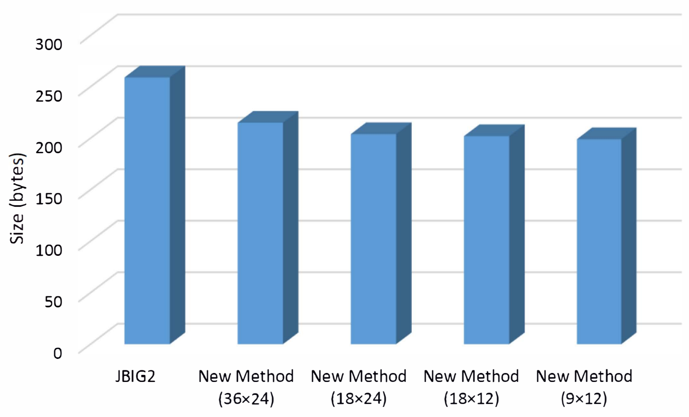

Fig. 16. Compressed file sizes (the New Method vs. JBIG2).

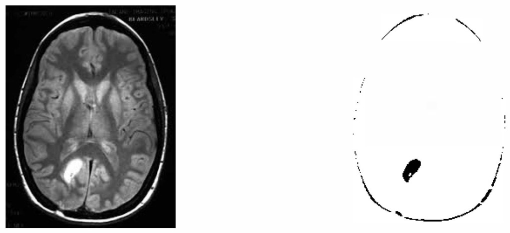

Fig. 17. MRI image of a brain tumor (left); the binary image (right).

block size achieved improvement over JBIG2 as shown in Fig. 16. The block size of $9 \times 12$ provides the best improvement (up to 23%) over the JBIG2 standard codec [24]. In addition, we applied our method on compression of medical images. Binary images can be very useful in medical imaging. For instance, the exact location of a tumor in the brain is extremely important and needs to be identified as shown in Fig. 17 (left) [25]. This image is an MRI of a brain with a tumor. Only for demonstration purpose, we converted the grayscale image of the MRI to a binary image (see Fig. 17 (right)). Two different block sizes were used for comparison. The compression result is shown in Fig. 18, where the block size $51 \times 39$ shows the best improvement over JBIG2 (by 22.44%).

## VI. Conclusion

We proposed a new method to improve our previous work on distance-coding of binary images, by run-length coding the most probable interval value independently from the rest of the interval values. Separate Huffman coding tables were used to code the run-lengths of the most probable interval versus other intervals. Consequently, this hybrid coding scheme allows a fractional bit to be assigned to the most probable interval value on average, as opposed to at least one bit per interval without run-length coding, thereby contributing to significant improvement on the compression efficiency. In the next step, we will consider several other coding methods such as Golomb code and its variants in compressing the run-lengths, as well as compare with other standard codec such as JBIG, and JPEG 2000.

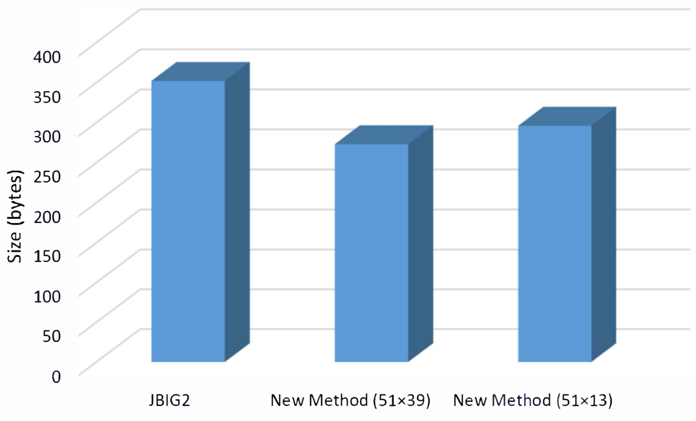

Fig. 18. Compressed file sizes (the New Method vs. JBIG2 codec).

## References

[1] J. Ascenso and F. Pereira, "Lossless compression of binary image descriptors for visual sensor networks," in *18th International Conference on Digital Signal Processing (DSP), 2013*, July 2013, pp. 1–8.

[2] A. Kazlouski and R. Sadykhov, "Plain objects detection in image based on a contour tracing algorithm in a binary image," in *IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, June 2014, pp. 242–248.

[3] J. H. Jaseema Yasmin, M. M. Sathik, and S. Z. Beevi, "Robust segmentation algorithm using log edge detector for effective border detection of noisy skin lesions," in *International Conference on Computer, Communication and Electrical Technology (ICCCET)*, March 2011, pp. 60–65.

[4] J. Jai Jaganath Babu and G. Sudha, "Non-subsampled contourlet transform based image denoising in ultrasound thyroid images using adaptive binary morphological operations," *IET Computer Vision*, vol. 8, no. 6, pp. 718–728, 2014.

[5] S. Sharma and P. Khanna, "Roi segmentation using local binary image," in *2013 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Nov 2013, pp. 136–141.

[6] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sept 1952.

[7] W.-W. Lu and M. Gough, "A fast-adaptive huffman coding algorithm," *IEEE Transactions on Communications*, vol. 41, no. 4, pp. 535–538, Apr 1993.

[8] S. Golomb, "Run-length encodings," *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 399–401, Jul 1966.

[9] H. Tanaka and A. Leon-Garcia, "Efficient run-length encodings," *IEEE Transactions on Information Theory*, vol. 28, no. 6, pp. 880–890, Nov 1982.

[10] J. Langdon, G.G. and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Transactions on Communications*, vol. 29, no. 6, pp. 858–867, Jun 1981.

[11] S. Zahir and M. Naqvi, "A new rectangular partitioning based lossless binary image compression scheme," in *2005 Canadian Conference on Electrical and Computer Engineering*, May 2005, pp. 281–285.

[12] L. Zhou and S. Zahir, "A new efficient algorithm for lossless binary image compression," in *Canadian Conference on Electrical and Computer Engineering, CCECE'06*, May 2006, pp. 1427–1431.

[13] I. 11544, "Information technology - coded representation of picture and audio information - progressive bi-level image compression," 1993.

[14] I. 14492, "Information technology - lossy/lossless coding of bilevel images," 2001.

[15] R. Raguram, M. Marcellin, and A. Bilgin, "Improved resolution scalability for bilevel image data in JPEG2000," *IEEE Transactions on Image Processing*, vol. 18, no. 4, pp. 774–782, April 2009.

[16] S. Deorowicz, "Second step algorithms in the burrows-wheeler com-

pression algorithm," *Software Practice and Experience*, vol. 32, no. 2, pp. 99–111, Feb 2002.

[17] T. Gagie and G. Manzini, "Move-to-front, distance coding, and inversion frequencies revisited," in *the 18th Annual Conference on Combinatorial Pattern Matching*, 2007, pp. 71–82.

[18] D. Adjeroh, T. Bell, and A. Mukherjee, *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*, 2008.

[19] W. Pan *et al.*, "A new method for compressing quality-controlled weather radar data by exploiting blankout markers due to thresholding opera-tions," in *AMS 27th Conference on International Interactive Information and Processing Systems (IIPS'11) for Meteorology, Oceanography, and Hydrology*, 2011.

[20] A. Liaghati and W. Pan, "An adaptive interval generation method for efficient distance coding of binary images," in *3rd International Conference on Image, Vision and Computing*, Sep. 2014.

[21] Video test media. [Online]. Available: http://media. xiph.org/video/derf/.

[22] S. Gerke, S. Singh, A. Linnemann, and P. Ndjiki-Nya, "Unsupervised color classifier training for soccer player detection," in *Visual Communications and Image Processing (VCIP), 2013*, Nov. 2013, pp. 1–5.

[23] Germany vs argentina 29 min offside goal. [Online]. Available: https://www.youtube.com/watch?v=NDEq44IEtSY

[24] jbig2 encoder. [Online]. Available: https://github.com/agl/jbig2enc.

[25] S. Beardsley. My journey living with a brain tumor. [Online]. Available: https://www.sandybeardsley.com