2019 Global Conference for Advancement in Technology (GCAT)
Bangalore, India. Oct 18-20, 2019

Image Compression using Run Length Encoding and its Optimisation

Amit Birajdar
Computer Engineering
MPSTME , NMIMS
Mumbai , India
amitbirajdar7@gmail.com

Harsh Agarwal
Computer Engineering
MPSTME , NMIMS
Mumbai , India
harsh30199@gmail.com

Manan Bolia
Computer Engineering
MPSTME , NMIMS
Mumbai , India
mananbolia14@gmail.com

Vedang Gupte
Computer Engineering
MPSTME , NMIMS
Mumbai , India
vedanggupte@gmail.com

***Abstract*— Images are among the most common and popular representations of data. Digital images are used for professional and personal use ranging from official documents to social media. Thus, any Organization or individual needs to store and share a large number of images. One of the most common issues associated with using images is the potentially large file-size of the image. Advancements in image acquisition technology and an increase in the popularity of digital content means that images now have very high resolutions and high quality, inevitably leading to an increase in size. Image compression has become one of the most important parts of image processing these days due to this. The goal is to achieve the least size possible for an image while not compromising on the quality of the image, that gives us the perfect balance. Therefore, to achieve this perfect balance many compression techniques have been devised and it is not possible to pinpoint the best one because it is really dependent on the type of image to be compressed. So here we are going to elaborate on converting images into binary images and the Run length Encoding (RLE) algorithm used for compressing binary images. Now, RLE is itself a very effective and simple approach for compression of images but, sometimes, the size of an image actually increases after RLE algorithm is applied to the image and this is one of the major drawbacks of RLE. In this research paper we are going to propose an extension or maybe an upgradation to RLE method which will ensure that the size of an image never exceeds beyond its original size, even in the worst possible scenario**

Keywords*— *Image Compression, Compression Ratio, Run Length Encoding (RLE), Selective Value Count (SVC), Optimal Thresholding

I. Introduction

Image Compression is a fundamental part of image processing. High quality images are often large in size and thus transmission of these images is slower and requires more bandwidth. A large file also requires more storage space on servers and thus limits the number of images that can be stored. Compression and be either lossy or lossless. Lossless compression aims to reduce the file size as much as possible without eliminating any of the detail in the image. When the file is decompressed, it gives back the exact original image without any degradation in quality. RLE is a very simple form of lossless data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original sequence. In this paper we have discussed on how to implement RLE with an algorithm. We then take a look at the drawbacks of conventional implementation of RLE along with the solutions to these problems and their algorithms. Furthermore, we have implemented the algorithms on different types of images and compared the results with the conventional RLE method to demonstrate the effectiveness of these new approaches.

II. Run Length Encoding

Run Length Encoding is a type of lossless compression method. We have used it to reduce the size of the image matrix by removing redundancy of repeated intensity values in an image by counting the runs of repeated values in the image. Then the code matrix is created where each intensity value is followed by the frequency of its sequential occurrence. This is most useful when data contains multiple runs of repeated values. Text documents are compressed easily and efficiently using this technique.

A. Compression :

1. Read input image
2. Encode the image by scanning each row until end of row is reached-
    a. For every set of repeating intensity value in image row, add an entry in encoded matrix at i^{th} position
    b. Count the number of consecutive instances of intensity value, enter this value in $(i+1)^{th}$ position
    c. If end of row is reached, Stop
3. Repeat for every row
4. Store the compressed image and size of each encoded row.

B. Decompression :

1. Accept the compressed image and the size of each row of image.
2. Repeat for every row element by element-
    a. For every odd position I of row, store the element as intensity value n
    b. Read next element i+1, which depicts the frequency of intensity value n, store value as m
    c. Copy intensity value n into decoded matrix m number of times.
    d. If row ends, move to next row
3. Display or store the decompressed image.

III. Drawbacks Of RLE

1. The compression ratio is dependent on the orientation of the image. Conventional RLE scans the image row wise. It will apply the same method on the images with landscape orientation as well as portrait orientation. It

978-1-7281-3694-3/19/$31.00 ©2019 IEEE

works on the images with landscape orientation and give a good output but it will not give good output on the images with portrait orientation. This is because in RLE, the number of rows in encoded image stay equal to the number of rows in original image, while the number of columns vary. Since in portrait mode the number of rows is greater than the number of columns, the size of the image can increase after applying RLE.

2. If the image is such that each line consists of rapidly alternating pixels of intensity 0 and 1, it will generate an extra column for each pixel to store its frequency. Thus, the final compressed image can potentially grow to twice the size of the original image.
3. RLE compresses runs of data but in case of color images there are hardly any consecutive pixels with same intensity value, therefore it gets difficult to compress these images.
4. In some images, there are too many intensity changes in the row while way lesser in column, so it is better to encode the image using columns instead of rows.

We will discuss in upcoming sections possible solution to each of the drawbacks mentioned above.

IV. Automatic Or Optimal Thresholding

Automatic thresholding considers the valley points in the histogram of the image to suitably threshold the image so that no data is lost during conversion from grayscale or color into binary. We can use automatic thresholding to deal with converting images to binary images.

Algorithm:

1. Calculate the smallest and the largest intensity values in the image and set initial Threshold value at T = (min + max) * 0.5
2. Segment image into two groups using T as group G1 as intensity values less than T and group G2 as intensity values greater than or equal to T
3. Calculate average intensity value of each group G1 and G2 separately as m1 and m2 respectively.
4. Compute new threshold value T1 as arithmetic mean of m1 and m2.
5. Repeat steps 2 through 4 as T = T1 until T ≈ T1
6. All pixels with intensity value less than T are assigned value 0 and pixels with intensity value greater than or equal to T are assigned value 1.

RLE Compression Algorithm:

1. Read input image
2. If image has more than one plane (e.g. red-green-blue), convert it to grey image
3. Use automatic thresholding to calculate optimum threshold value.
4. Encode the image by scanning each row until end of row is reached-
    a. For every set of repeating intensity value in image row, add an entry in encoded matrix at i^{th} position
    b. Count the number of consecutive instances of intensity value, enter this value in $(i+1)^{th}$ position
    c. If end of row is reached, Stop
5. Repeat for every row.
6. Store the compressed image and size of each encoded row.

V. Orientation RLE

In this we check the orientation of image whether it is landscape or portrait image, in case of landscape images no change is made while in case of portrait images, transpose of the image is taken to convert them into landscape images and then they are encoded.

A. Compression :

1. Read input image
2. If image has more than one plane (e.g. red-green-blue), convert it to grey image
3. Use automatic thresholding to calculate optimum threshold value.
4. If number of rows >= number of columns, then take transpose of image matrix and set flag
5. Encode the image by scanning each row until end of row is reached-
    a. For every element matrix, add an entry in encoded matrix at ith position
    b. Count the number of consecutive instances of element, enter this value in (i+1) th position
    c. If end of row is reached, Stop
6. Repeat for every row
7. Store the compressed image and size of each encoded row.

B. Decompression:

1. Accept the compressed image and the size of each row of image.
2. Repeat for every row element by element-
    a. For every odd position I of row, store the element as intensity value n
    b. Read next element i+1, which depicts the frequency of intensity value n, store value as m
    c. Copy intensity value n into decoded matrix m number of times.
    d. If row ends, move to next row
3. If flag is set, transpose decoded image
4. Display or store the decompressed image.

VI. Sign Change RLE

In this approach, we assume intensity value 0 as negative (-) sign and intensity value 1 as positive (+) sign and we calculate the no of times in each row neighboring pixels are of different signs for all the rows and calculate their sum as rowchange. Similarly, for column is done and called columnchange and using some conditions whether image has to be rotated or not is decided.

A. Compression :

1. Read input image
2. If image has more than one plane (e.g. rgb), convert it to grey image
3. Use automatic thresholding to calculate optimum threshold value.

4. For each row in the image, store the number of sign changes in neighboring intensity values (0 to 1 or 1 to 0) a
5. Calculate the sum of sign change values of all rows as rowchange (RSC).
6. For each column in the image, store the number of sign changes in neighboring intensity values (0 to 1 or 1 to 0)
7. Calculate the sum of sign change values of all columns as columnchange (CSC)
8. Compare RSC with CSC using (RSC – CSC)/RSC and store as factor (μ). For landscape image if μ>0.75 then take transpose of image and set flag. For portrait image if μ>0.5 then take transpose of image and set flag
9. Encode the image by scanning each row until end of row is reached-
    a. For every element in image matrix, add an entry in encoded matrix at ith position
    b. Count the number of consecutive instances of element, enter this value in (i+1) position
    c. If end of row is reached, Stop
10. Repeat for every row.
11. Store the compressed image and size of each encoded row.

B. Decompression

1. Accept the compressed image and the size of each row of image.
2. Repeat for every row element by element-
    a. For every odd position I of row, store the element as intensity value n
    b. Read next element i+1, which depicts the frequency of intensity value n, store value as m
    c. Copy intensity value n into decoded matrix m number of times.
    d. If row ends, move to next row
3. If flag is set, transpose decoded image
4. Display or store the decompressed image.

VII. Selective Value Count RLE

Selective Value Count (SVC) is a method in which only when there is more than one cumulative pixel with same intensity value are converted to intensity value, frequency pair. In case there is single occurrence of an intensity value then only the intensity value is stored and not its frequency.

For example: 111101 is the row to be encoded, then RLE encodes it as 140111, while SVC RLE encodes it as 1401. Therefore, saving two less cells of data.

A. Compression :

1. Read input image
2. If image has more than one plane (e.g. rgb), convert it to grey image
3. Use automatic thresholding to calculate optimum threshold value.
4. Encode the image by scanning each row until end of row is reached-
    a. Set Counter to 0
    b. Let selected pixel be i, let j = (i+1) Element.
    c. If i = j, then Count the number of consecutive instances of element, enter this value in (i+1) position and in ith position store intensity value. Go to step e
    d. If i ≠ j, then insert intensity value at i position and move to next pixel
    e. End of row reached, Stop
5. Repeat for each row
6. Store the compressed image and size of each encoded row.

B. Decompression :

1. Accept the compressed image and the size of each row of image.
2. Repeat for every row element by element-
    a. For position i check whether i+1th value is greater than 1, if no go to c
    b. Store value at i+1 th position as m and value at ith position as n. Go to d
    c. Store value at ith position as n and m = 1
    d. Repeat intensity value n, m times in the decoded image matrix
    e. If end of row, move to next row.
3. Display or store the decoded image.

VIII. Optimised RLE

Optimized RLE is the combination of all the solutions decided above to create an RLE algorithm which gives optimum compression ratio in each and every scenario. In this we implement automatic thresholding, Orientation, Sign Change and Selective Value Count methods to form an optimal algorithm.

A. Compression :

1. Read input image
2. If image has more than one plane (e.g. rgb), convert it to grey image
3. Use automatic thresholding to calculate optimum threshold value.
4. If number of rows >= number of columns, then take transpose of image matrix and set flag1
5. For each row in the image, store the number of sign changes in neighboring intensity values (0 to 1 or 1 to 0)
6. Calculate the sum of sign change values of all rows as rowchange (RSC).
7. For each column in the image, store the number of sign changes in neighboring intensity values (0 to 1 or 1 to 0)
8. Calculate the sum of sign change values of all columns as columnchange (CSC)
9. Compare RSC with CSC using (RSC – CSC)/RSC and store as factor (μ). For landscape image if μ>0.75 then take transpose of image and set flag. For portrait image if μ>0.5 then take transpose of image and set flag2
10. Encode the image by scanning each row until end of row is reached-
    a. Set Counter to 0

b. Let selected pixel be i, let j = (i+1) Element.
c. If i = j, then Count the number of consecutive instances of element, enter this value in (i+1) position and in ith position store intensity value. Go to step e
d. If i ≠ j, then insert intensity value at i position and move to next pixel
e. End of row reached, Stop

11. Repeat for each row
12. Store the compressed image and size of each encoded row.

B. Decompression :

1. Accept the compressed image and the size of each row of image.
2. Repeat for every row element by element-
   a. For position i check whether i+1th value is greater than 1, if no go to c
   b. Store value at i+1 th position as m and value at ith position as n. Go to d
   c. Store value at ith position as n and m = 1
   d. Repeat intensity value n, m times in the decoded image matrix
   e. If end of row, move to next row.
3. Perform XOR operation on flag1 and flag2, if 1 then perform transpose of the image.
4. Display or store the decoded image.

IX. INPUT DATA

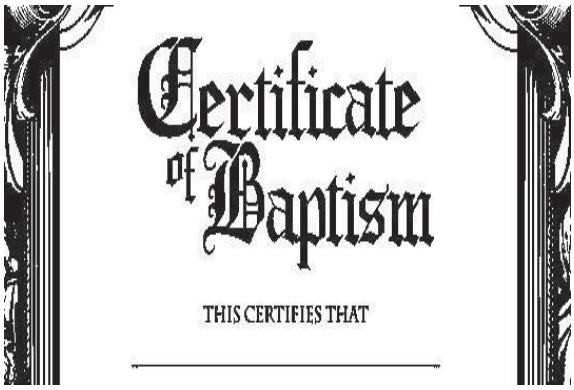

Fig. 1. Input Data 1

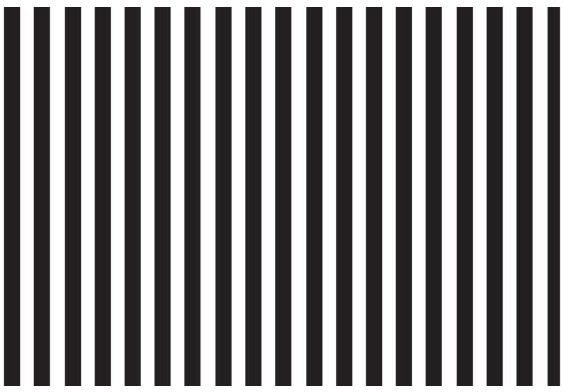

Fig. 2. Input Data 2

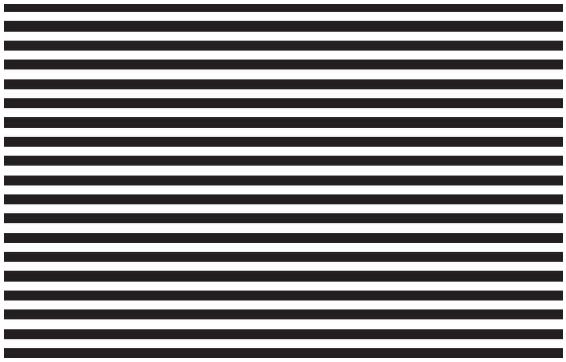

Fig. 3. Input Data 3

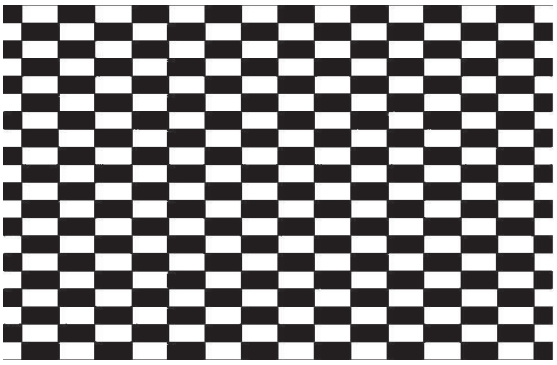

Fig. 4. Input Data 4


2018 STEAM INNOVATORS CAMP

I. Registration Form

II. Consent to Participate in the 2018 STEAM Innovators Camp

Fig. 5. Input Data

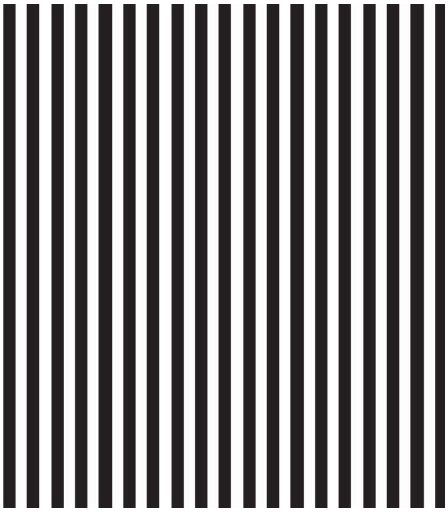

Fig. 6. Input Data 6

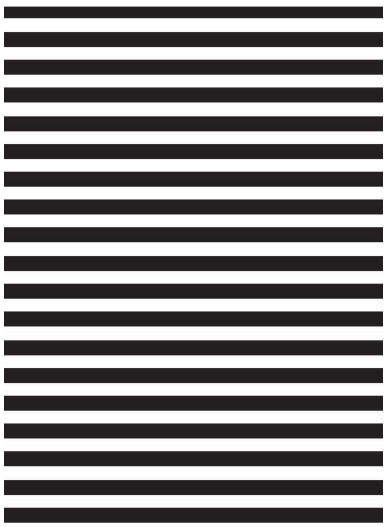

Fig. 7. Input Data 7

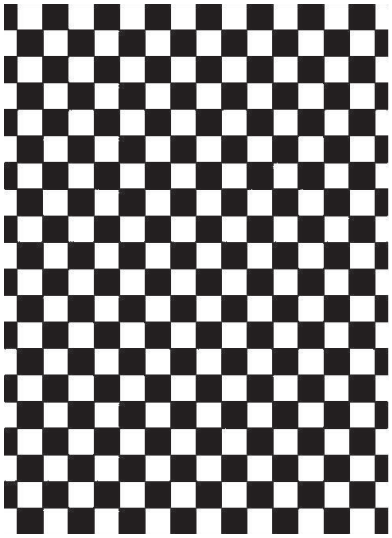

Fig. 8. Input Data 8

X. Observation Table

TABLE I. RLE

Image No	Input Image Size (kb) (I)	Encoded Image Size (kb) (E)	Compression Ratio (1-(E/I))
1	27	19	0.296
2	27	20	0.259
3	41	2	0.951
4	43	25	0.418
5	151	157	-0.039
6	42	31	0.261
7	26	2	0.923
8	43	25	0.418

TABLE II. SIGN CHANGE AND ORIENTATION RLE

Image No	Input Image Size (kb) (I)	Encoded Image Size (kb) (E)	Compression Ratio (1-(E/I))
1	27	19	0.296
2	27	2.44	0.909
93	41	1.38	0.966
4	43	19.8	0.539
5	151	95.1	0.370
6	42	1.38	0.967
7	26	2.44	0.906
8	43	19.8	0.539

TABLE III. SELECTIVE COUNT RLE

Image No	Input Image Size (kb) (I)	Encoded Image Size (kb) (E)	Compression Ratio (1-(E/I))
1	27	18.4	0.318
2	27	18.5	0.314
3	41	1.38	0.966
4	43	25.8	0.4
5	151	150.5	0.003
6	42	29.7	0.292
7	26	2.44	0.906
8	43	25.8	0.4

TABLE IV. OPTIMISED RLE

Image No	Input Image Size (kb) (I)	Encoded Image Size (kb) (E)	Compression Ratio (1-(E/I))
1	27	18.4	0.318
2	27	2.44	0.909
3	41	1.38	0.966
4	43	18.8	0.562
5	151	92.3	0.388
6	42	1.38	0.967
7	26	2.44	0.906
8	43	18.8	0.562

XI. Graphs and Inferences

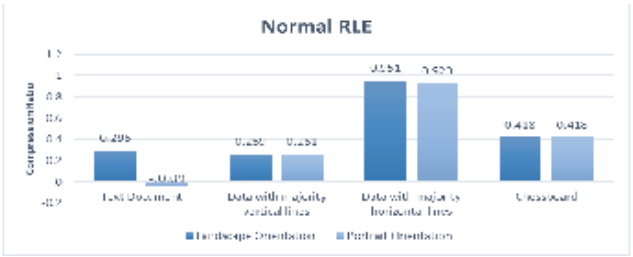

Fig. 9. Normal RLE

1. For a general text document, landscape orientation gives better compression as compared to the portrait orientation. In fact, the portrait orientation does not compress the file, i.e. size of encoded image is greater than the size of input image file.
2. For documents with majority horizontal and vertical lines, compression ratio for landscape orientation is observed to be slightly better than portrait orientation.
3. In case of the chessboard pattern, we observe that there is no effect of the orientation. In either case, compression ratio is the same.

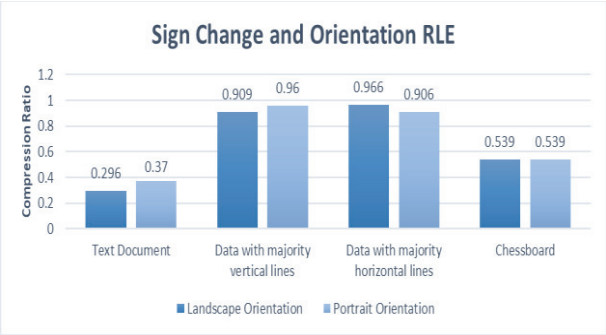

Fig. 10. Sign Change and Orientation RLE

Sign change gives us the orientation of the density of data present in the document. The text document can have data either horizontally or vertically dense. Using the knowledge of the orientation of data, we apply different compression techniques to check which method gives the best output. From the above chart, we can observe that for a document with majority vertical data, portrait orientation gives better results. For a document with majority horizontal data, landscape orientation gives better results. In case of the chessboard pattern, since data is equally distributed horizontally and vertically, therefore the orientation does not matter. Any orientation would give the same amount of compression.

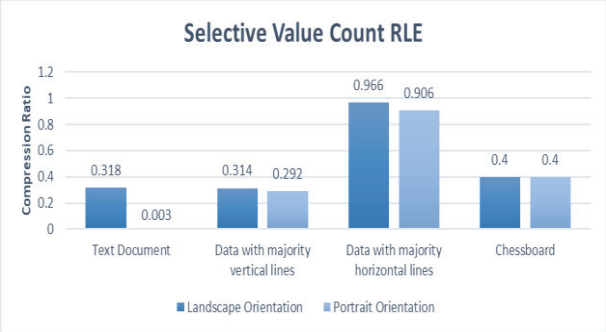

Fig. 11. Selective Value Count RLE

In this method, as we discussed, frequency of intensity values is taken into consideration. This method proves to be very effective in compressing document image files. As we observe in the graph above, compression of a text document in portrait orientation gives an insignificant amount of compression. Whereas, in case of a document having majority horizontal lines, landscape orientation gives very high compression ratio. Compression in case of chessboard format is equal in either orientations.

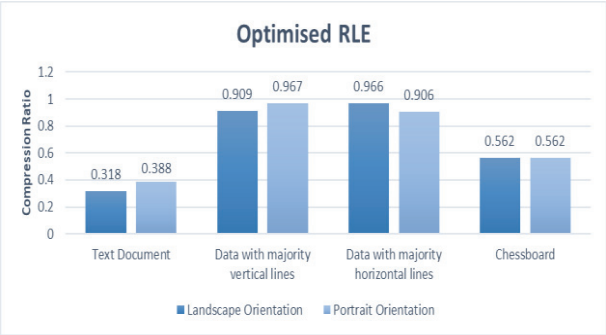

Fig. 12. Optimised RLE

As we can see in the graph that in all scenarios we get best compression ratio in the Optimised RLE Algorithm , with the traditional RLE , compression ratio for portrait document was negative (-0.039) while through Optimised RLE , compression ratio for the above is positive (0.388) , which is almost 10 times better than traditional algorithm.

XII. Conclusion

In this paper we observed that Run Length Encoding is powerful and simple algorithm but only in the most ideal cases, it cannot deal with complex images such as images of forms with lots of data in it, for example an image which has a lot of 0s and 1s in alternate positions consecutively, images like these are more likely to expand in size after going through Run Length Encoding. Now, even if the compression is not possible due to the multiple consecutive 0s and 1s, at least the size of the image should not increase, and this is exactly what we have achieved through selective value count method and to finally achieve the best compression possible we have combined it with the Orientation and Sign Change RLE approach which checks the image and tells us the best orientation to apply SVC RLE in, this gives us the most suitable direction to apply the algorithm in, thus making the algorithm intelligent. It also gives us a more efficient method to compress the image making the algorithm more robust. So, a robust algorithm along with intelligence ensures the most efficient compression of images through Run Length Encoding algorithm as the base.

References

[1] Image Compression Using Run Length Encoding (RLE) Kamalpreet Kaur1 , Jyoti Saxena2 and Sukhjinder Singh3 1Research Scholar, 2 Professor and 3Assistant Professor 1,2,3Department of Electronics & Communication Engineering Giani Zail Singh Campus College of Engineering & Technology, Bathinda-151001 (Punjab)

[2] Al-laham, M., Emary, I.: Comparative study between various algorithms of data compression techniques. IJCSNS, International Journal of Computer Science and Network Security 7(4), 284–290 (2007)

[3] Modified Run-Length Encoding Method and Distance Algorithm to Classify Run-Length Encoded Binary Data - T. Kathirvalavakumar and R. Palaniappan

[4] Improved Run Length Encoding Scheme For Efficient Compression Data RateS. Sarika*, S. Srilali**(Department ofElectronics and CommunicationEngineering)(Swarnandhra College of Engineering and Technology)