CHAPTER 4

IMPROVED BLOCK BASED SEGMENTATION ALGORITHM FOR COMPRESSION OF COMPOUND IMAGES

4.1 INTRODUCTION

Internet, fax, mobile phones etc are the most common mediums of communication in use nowadays. The information used in these mediums are in digital format. Transmission of these digital forms of documents is done in a fraction of a second. To avoid expenses and delay, high compression and bit rates are required during transmission. Thus, data compression becomes unavoidable.

Electronic document images are often a mixture of data types such as text, background, and foreground. These types of images are named as compound images. Compound images are one of the best way for representing information. Compound images are images with different formats that contains both palletized regions which have text or graphics and uninterrupted tone regions. Examples for typical compound images are hybrid documents, captured screen contents, slides and web-pages given by Maheswari & Radha (2011).

In this work a Novel block based segmentation algorithm for compression of compound image has been implemented. In this method, at first the pictorial blocks and the text blocks are separated from the compound image by using text detection using MSER based method.
The pictorial blocks are compressed using SVD based compression technique and the text blocks are mapped to primary colors using color quantization algorithm and the resulting indexed values are then compressed using lossless Improved Huffman coding technique.

Then the compressed blocks are merged together and the encryption algorithm has been performed which ensures the transmission is fast and highly secured.

4.2 PREVIOUS ARTS

Compound images must be compressed and transmitted in a secure way. Especially with the popularity of cloud computing, compound images must be displayed on remote clients, wireless projectors and thin clients. Some clients are not able to provide them from files directly. Compression and transmission of compound images renders a common solution to such clients. In this solution Lan et al (2010) proposed an effective compression of compound images has become a widespread and vital problem.

Cheng et al (2002) proposed a method by exploiting spatial correlation. In the absence of data compression algorithms, it is absolutely impossible to transmit a large volume of data over the up to date bandwidth-constrained networks in real time.

Compressing compound images with a single algorithm that simultaneously meets the requirements for text, image and graphics has been elusive and thus a new algorithm that can expertly reduce the file size without degrading the quality is required. The compound image compression performance essentially depends on the segmentation result.
A segmentation process is used where regions of related data types are grouped as one. After successful segmentation, existing compression techniques that best match each data type can be used to achieve best compression results. Segmentation algorithms for compound image compression are normally grouped as block based, object based and layer based segmentation. Each and every method has its own merits and demerits. Most of the recent researches in this field are mainly based on either layer based or block based.

In object based method, a page is divided into regions where each region follows exact object precincts. An object may correspond to a photograph, a graphical object, a letter, etc. Complexity is the major drawback of this method. In layer-based method, a page is alienated into rectangular layers. The standard three layers mixed raster content representation plays a major role in the most layered coding algorithms Proposed by Ding et al (2006). Some traditional compressors like DjVu, Digipaper proposed by Felzenszwalb & Rucklidge (1999) and JPEG 2000 proposed by Sharpe & Buckley (2000) are also available.

In block-based method, a page is divided into rectangular blocks where each block follows precise object restrictions. Simplified segmentation, better match between region boundaries and the compression algorithms and the lack of redundancy are the advantages of this approach.

Block-based approaches for multifaceted descriptions are also studied for their low complexity. Block-based approaches segment non-overlapping blocks of pixels into different classes and compact each class differently according to its distinctiveness. Block based approaches in the coding structures is very flexible. Said & Drukarev (1999) proposed a simple
block-based scheme which compresses text blocks using JPEG-LS and picture blocks using JPEG.

Most existing coding algorithms with desirable features (e.g. resolution scalability, rate-distortion optimization) can be easily applied when the image is structured into blocks instead of layers proposed by Bottou et al (1998). Despite its advantages, the security of the compressed image is an important issue that has been considered by researchers over the past. Security of the compressed images is improved by using various encryption and decryption techniques.

Compressing compound images with a single algorithm that concurrently meets the requirements for text, image and graphics has been elusive and thus, requires new algorithms that can competently reduce the file size without corrupting the quality. The compound image compression performance mainly depends on the segmentation outcome. Regions of similar data types are grouped together in the segmentation process. After successful segmentation, the existing techniques that best suit to achieve the best compression results. Segmentation algorithms for compound image compression are normally grouped as block based, object based and layer based segmentation. Each and every method has its own merits and demerits. Most of the recent researches in this field are mainly based on either layer based or block based.

Maheswari & Radha (2011) have proposed a model that dealt with a comparison of layer and block based classification methods. For secure transmission, an encryption algorithm was used to encrypt the compressed image. Experimental results were conducted to analyze the performance of the two compressors.
Ding et al (2006) have proposed a novel block based fast compression (BFC) algorithm for compound images that contain graphics, text and natural images. The images were divided to blocks which were classified into four different types – smooth blocks, text blocks, hybrid blocks and picture blocks with a fast and effective block-based classification algorithm.

Four different coding algorithms were carefully designed for each block type according to their different statistical properties to maximize the compression performance. Simulations showed that the BFC algorithm they proposed has a much lower complexity than DjVu with a significantly better visual quality at high bit rate and it also outperformed the popular lossy image coding method JPEG.

Talukder & Harada (2007) have proposed a Discrete Wavelet Transform for Image Compression and A Model of Parallel Image Compression Scheme for Formal Verification. They used a model of the scheme of verification of parallelizing the compression.

It is well known that wavelet transform is especially useful to transform images. They applied it twice: first on rows, second on columns. Upon this, the image matrix was deinterleaved and recursively transformed each sub-band individually further for the compression.

Lin & Pengwei (2005) have proposed a compound image compression algorithm for real-time applications of computer screen image transmission. It is called Shape Primitive Extraction and Coding (SPEC). Real-time image transmission requires that the compression algorithm should not only achieve high compression ratio, but also have low complexity and provide excellent visual quality. SPEC first segmented a compound image into text/graphic pixels and pictorial pixels and then compressed the text/graphic
pixels with a new lossless coding algorithm and the pictorial pixels with the standard lossy JPEG, respectively.

The segmentation first classified image blocks into picture and text/graphics blocks using color number thresholding. Then, it extracted shape primitives of text/graphics from the picture blocks. A dynamic color palette that tracks recent text/graphic colors was used to separate small shape primitives of text/graphics from pictorial pixels.

Shape primitives were also extracted from text/graphics blocks. All shape primitives from both block types were compressed by using a combined shape-based and palette-based coding algorithm. The final bitstream of lossless coding was fed into an LZW coder. Experimental results showed that the SPEC has very low complexity and provides visually lossless quality while keeping competitive compression ratios.

Ding et al (2007) have proposed an efficient compound image compression approach based on H.264/AVC intra coding. The text blocks were distinguished from the picture blocks and compressed in a new coding mode. In particular, the text blocks were represented by base colors and index map in spatial domain. Menachem (2009) proposed an Huffman coding method for text, images and music characters in DNA.

A color quantization algorithm optimized for compression was designed to generate this representation. As for the entropy coding of text blocks, a structure-aware context-based arithmetic coder was developed. The mode selection algorithm based on rate distortion optimization was used to select the text blocks along with H.264/AVC intra modes, which could adapt to the targeted bit-rate. Experimental results showed that the proposed scheme
could achieve 2.8 dB gain on average for compound images when compared with H.264/AVC intra coding.

Daeme and Vincent (2000) have proposed the cipher Rijndael. They presented the mathematical basis necessary for understanding the specifications followed by the design rationale and the description itself. Subsequently, the implementation aspects of the cipher and its inverse were treated.

This was followed by the motivations of all design choices and the treatment of the resistance against all known types of attacks. They gave security claims and goals, the advantages and limitations of the cipher, ways how it could be extended and how it could be used for functionality other than block encryption/decryption.

Jagadish and Lohit (2010) have proposed a lossless method of image compression and decompression using a simple coding technique called Huffman coding. This technique was simple in implementation and utilized less memory. A software algorithm had been developed and implemented to compress and decompress the given image using the Huffman coding technique in a MATLAB platform.

Tony et al (2005) have proposed a new compound image compression algorithm based on Shape Primitive Extraction and Coding (SPEC). The SPEC first segmented a compound image into text/graphic pixels and pictorial pixels by extracting the shape primitives of text/graphics. Then all the shape primitives were losslessly compressed with a combined shape-based and palette based coding algorithm.

The remaining pictorial pixels were coded with JPEG2000. Experimental results showed that the SPEC has very low complexity and
provided visually lossless quality while yielding competitive compression ratios.

Li & Lei (2001) have proposed a novel block-based segmentation and adaptive coding (BSAC) algorithm for visually lossless compression of scanned documents that contain not only photographic images but also text and graphic images.

For such a compound image source, they structured the image into non overlapping blocks and classified each block into four different classes based on the empirical statistics within the block. Different coding strategies were applied to different classes in order to achieve the very best compression performance. Their new block - based image coder was able to provide visually lossless compression of scanned documents at the bit rate of around 1-1.5bpp with modest computational complexity and very low memory requirement. Gai et al (2013) propose a new multiscale texture classifier which uses features extracted from the sub-bands of the RQWT decomposition in the transform domain.

MSERs pruning algorithm by Carlos et al (2011) contains two steps, reduction of linear segments by maximizing the border energy function and hierarchical filtering with a cascade of filters.

Neumann & Matas (2011) proposed a MSER++ based text detection method which exploits rather complicated features. Neumann & Matas (2012) also proposed a two stage algorithm for extremal regions(ERs) pruning with the exhaustive search strategy.

The clustering based method by Pan et al (2011) clusters character candidates into a tree using the minimum spanning algorithm with a learned distance metric proposed by Yin & Liu (2009).
Text candidates construction has been done by two approaches, rule based and clustering based approach.

The Previous works use different methods for estimating the probability of MSERs. For large number of repeating MSERs they use relevant features (cascading filters and incrementally computable descriptors) in pruning. Another problem for connected component method and hybrid method is the absence of an effective text candidates construction algorithm. The rule based methods require hand-tuned parameters, while the clustering method is complicated by post-processing stage, where one has to specify a rather complicated energy model.

Encryption in frequency domain can be classified into block based and image based. Block based approach uses DCT and DFT. The limitation is only the spatial correlation inside the 2-D block is considered and the correlation from the pixels of the neighbouring blocks is neglected. So it is impossible to decorrelate the blocks at their boundaries.

The discrete cosine transform is inflexible i.e., not efficient for binary images. So, image based transformation method is preferred in this work which transforms the whole image as a unit. The image based transformation provides high flexibility. The DWT does not require dividing the input into non-overlapping blocks

4.3 PROPOSED METHOD

A Novel block based segmentation algorithm for compression of compound image has been implemented. In this method, at first the pictorial blocks and the text blocks are separated from the compound image by using text detection using MSER based method.
The pictorial blocks are compressed using SVD based compression technique and the text blocks are mapped to primary colors using color quantization algorithm and the resulting indexed values are then compressed using lossless Improved Huffman coding technique.

Then the compressed blocks are merged together and the encryption algorithm has been performed which ensures the transmission is fast and highly secured. The block diagram for the proposed scheme was shown in Figure 4.1.

![Figure 4.1 Block diagram for proposed scheme](image_url)

### 4.3.1 Text Detection Using MSER Method

By using an MSER text detection method the text from the compound images has been detected and a separate compression method has been performed for text pixels. The proposed text detection method includes the following stages:
1. **Character extraction.** Character is extracted from the compound images using MSERs algorithm by minimizing regularized variations. Most of the repeating components are removed by the proposed MSERs pruning algorithm.

2. **Text construction.** By using metric learning algorithm Distance weights and clustering threshold are learned simultaneously. Characters are clustered into text by single-link clustering algorithm using the learned parameters.

3. **Text elimination.** The probabilities of text corresponding to non-texts are estimated using the character classifier and text with high non-text probabilities are removed.

4. **Text classification.** By using the text classifier the true text are identified. An Ada Boost classifier proposed by Yin (2012) is used to decide whether an pixel corresponding to the true text or not.

### 4.3.1.1 Character candidates extraction

#### 4.3.1.1.1 Pruning algorithm overview

Repeating components are the major pitfall when the MSER algorithm is applied as a character segmentation algorithm. Figure 4.2 (a) shows that there are fourteen MSERs are detected for the word “PACT” but only four of that detected words are really needed to us. The hierarchical structure of MSERs are quite useful for designing a pruning algorithm. Once the parent node is known to be a character it will be safe to remove the children node and vice versa. By applying this kind of parent-children elimination the MSERs tree is pruned and all characters are preserved after the elimination.
As an example, Figure 4.2(e) shows that all the desired characters can be extracted by applying Pruning algorithm to the MSERs tree as shown in Figure 4.2. However, it can be computationally expensive to identify characters, which usually entails the computations of complex features.

Instead of identifying the character, simply we can choose any one character, i.e., more likely to be characters in a parent-children relationship. In text localization, real-word images are using efficiently pruned exhaustive search” Neuman (2011) says that the parent-children relationship is not sufficient to eliminate non character MSERs. The experimental results shows by using the proposed regularized variation scheme, the probability can be estimated very fast. If one child have multiple children in MSERs trees, we design two algorithms based on parent-children elimination.

i.e., Linear reduction and Tree Accumulation

The basic function of linear reduction algorithm is to remove the line segments in MSERs tree and then by using accumulation algorithm the repeated characters can be removed.

4.3.1.1.2 Variation and its regularization

An “extremal region” is a connected component of an image whose pixels have either higher or lower intensity than its outer boundary pixel. From
the whole image the extremal regions are extracted as a rooted tree. Infact an extremal region is also be a set of pixels. The variation of \( R_i \) is defined as in equation 4.1.

\[
V(R_i) = \frac{|R_{i+\Delta} - R_i|}{|R_i|}
\]  

(4.1)

where \( R_i \) is the extremal region.

The branch of the tree rooted at \( R_i \), \( B(R_i) = (R_i, R_{i+1}, \ldots, R_{i+\Delta}) \)

\(|R|\) denotes number of pixels

Matas et al (2002) in his work says that, If the variation is lower and more stable than its parent node \( R_{i-1} \) and the child node \( R_{i+1} \), the extremal region \( R_i \) is maximally stable extremal region. The size of the maximally stable extremal region remains virtually unchanged over a range of intensity levels.

If the MSERs tree with lower variations are more likely to be characters, there will be the possible strategy for parent-children elimination. ie. If the parent has the lowest variation it can be eliminated else the children can be eliminated. Since, MSERs corresponding to characters may not be necessarily have lowest variation, this strategy alone will not work.

Let \( V \) be the variation and \( a \) be the aspect ratio of a MSER, the aspect ratios of characters are expected to fall in \([amin, amax]\), the regularized variation can be defined as in equation 4.2.

\[
V = \begin{cases} 
V + \theta_1(a - amax) & \text{if } a > amax \\
V + \theta_2(amin - a) & \text{if } a < amin \\
V & \text{otherwise}
\end{cases}
\]
where $\theta_1$ and $\theta_2$ are penalty parameters, these parameters are set as $\theta_1 = 0.01$, $\theta_2 = 0.35$, $a_{\text{max}} = 1.2$ and $a_{\text{min}} = 0.3$.

### 4.3.1.1.3 Linear reduction

If MSERs has only one child, linear reduction algorithm is used. This algorithm chooses either parent or child with minimum variation and discards the other. This procedure is repeated for the whole tree. The detailed algorithm is given below.

```
Procedure LINEAR REDU(T)
    If nchildren[T]=0 then
        Return T
    Else if nchildren[T]=1 then
        c←LINEAR REDU(child[T])
        if var[T]≤var[c] then
            link children(T,children[c])
            return T
        else
            return c
        end if
    else
        :nchildren[T]≥2
        for each c ∈ children[T] do
            link children(T,LINEAR REDU(c))
        end for
        return T
    end if
end procedure
```
The procedure works as follows. Given a node \( t \), the procedure checks the number of children of \( t \); if \( t \) has no children, return \( t \) immediately; if \( t \) has only one child, get the root \( c \) of child tree by first applying the linear reduction procedure to the child tree; if \( t \) has a lower variation compared to \( c \), link \( c \)'s children to \( t \) and return \( t \); otherwise return \( c \); if \( t \) has more than one children, process these children using linear reduction and link the resulting trees to \( t \) before returning \( t \).

After applying linear reduction to the tree all linear segments are reduced and non leaf nodes have more than one children as shown in Figure 4.2(d).

4.3.1.1.4 Tree accumulation

If MSERs have more than one child, tree accumulation algorithm is used. The detailed algorithm is given below.

Procedure TREE ACCUMULAT(T)
If nchildren[T]≥2 then
    c← 0
    for each c є children[T] do
        c←c ∪ TREE-ACCUMULAT(c)
    end for
    if var[T]≤min-variance[c] then
        discard children(T)
        return T
    else
        return c
    end if
else  :nchildren[T]=0
return T
end if
end procedure

The algorithm works as follows. For a given node $t$, tree accumulation checks the number of $t$’s children; if $t$ has no children, return to $t$ immediately; if $t$ has more than two children, create an empty set $C$ and append the result by applying tree accumulation to $t$’s children to $C$; if one of the nodes in $C$ has a lower variation than $t$’s variation, return $C$, otherwise discard $t$’s children and return $t$.

After applying tree accumulation it is a set of disconnected nodes contains all desired characters as shown in Figure 4.2(e).

### 4.3.1.1.5 Complexity

The computations for each and every nodes in MSERs tree can be visit by linear reduction and tree accumulation algorithms effectively and do simple comparisons and pointer manipulations, thus the complexity is linear to the number of tree nodes. MSERs variations are already computed in MSER extraction process, the computational complexity of variation is mostly due to calculations of MSERs bounding rectangles, which are incrementally computable which was proposed by Neuman et al (2012).

### 4.3.1.2 Text candidates construction

Using single link clustering the character candidates are clustered together to construct text candidates. The clusters produced by single link clustering are elongated proposed by jain et al (1999) and it can be suitable for text candidate construction. Single link clustering is one of the types of hierarchical clustering.
In hierarchical clustering each data is treated as single cluster and all the clusters are merged together until all points are merged into a single cluster. If two clusters whose two closest members having the smallest distance are merged together in each step in single link clustering. If the distance between nearest clustered exceeds the threshold, the clustering process can be terminated. The remaining clusters of single link algorithm forms a hierarchical clustered tree. The weighted sum of features is used as a distance function in one algorithm.

Given two data points x, y let $X_{x,y}$ be the feature vector it characterize the similarity between x, y and the distance between x and y is defined as shown in equation 4.3.

$$D(x,y,W)=W^TX_{x,y} \quad (4.3)$$

where W is the feature weight vector together with threshold can be learned using the distance metric learning algorithm.

### 4.3.1.2.1 Feature space

The similarity between x and y can be determined by using feature vector $X_{x,y}$ the coordinates of the top left corner x’s bounding rectangle be $X_x$, $Y_x$, the height of the bounding rectangle is $H_x$, the width of the bounding rectangle is $W_x$. $S_x$ is the stroke width given by Yin et al (2012), $C_{1x}, C_{2x}, C_{3x}$ be the average three channel color values of X.

The features of feature vector $X_{x,y}$ are calculated by below equations.

1) Interval

$$\begin{align*}
\frac{\text{abs}(X_y - X_x - W_x)}{\text{max}(W_x, W_y)} & \quad \text{if } X_x < X_y \\
\frac{\text{abs}(X_x - X_y - W_y)}{\text{max}(W_x, W_y)} & \quad \text{otherwise}
\end{align*} \quad (4.4)$$

2) Width is given by

$$\frac{\text{abs}(W_x - W_y)}{\text{max}(W_x, W_y)} \quad (4.5)$$
and Height Differences is given by
\[
\frac{\text{abs}(H_x - H_y)}{\max(H_x, H_y)}
\]
(4.6)

3) Top alignment
\[
\text{Arcatan}\left(\frac{\text{abs}(Y_x - Y_y)}{\text{abs}\left(X_x + \frac{W_x}{2} - X_y - \frac{W_x}{2}\right)}\right)
\]
(4.7)

And the bottom alignment
\[
\text{Arcatan}\left(\frac{\text{abs}(Y_x + H_x - Y_y - H_y)}{\text{abs}\left(X_x + \frac{W_x}{2} - X_y - \frac{W_x}{2}\right)}\right)
\]
(4.8)

4) Color difference
\[
\frac{\sqrt{(C_{1x} - C_{1y})^2 + (C_{2x} - C_{2y})^2 + (C_{3x} - C_{3y})^2}}{255}
\]
(4.9)

5) Stroke width difference
\[
\frac{\text{abs}(S_x - S_y)}{\max(S_x, S_y)}
\]
(4.10)

4.3.1.2.2 Distance metric learning

Kumar et al (2008) proposed a metric algorithm in which it splits the whole data space into K partitions by clustering. A weighted Euclidean distance is learned within each cluster. It was having drawbacks that the learned metric corresponding to K clustering result and cannot be easily generalized to unobserved clusters.

This method compared their works with relative sense and often obtained without labelling objects. Besides learning from relative comparisons, metric learning is a much broader area with many efforts on learning from pairwise constraints. So we propose a metric learning from relative comparisons.

The important task of semi supervised clustering specifies the distance metric that satisfies the labels are constraints in supervised data, given
the clustering algorithm by Bilenko et al (2004), Xing et al (2002), Klein et al (2002). The metric learning is used to learn the distance function by minimizing the distance between point pairs in C while maximizing the distance between point pairs in M. Where C is the pairs of points in different clusters, M is the pair of points in same clusters.

In single link clustering the small clusters are merged together to form final resulting clusters. The final resulting clusters forms a binary cluster tree in which no single clusters have same two sub clusters. The termination threshold ($\epsilon$), follows the distance between sub clusters of each top level clusters are less or equal to $\epsilon$. The distance between data pairs in different top levels clusters are greater than $\epsilon$. This property of single level clustering is used to learn the distance function and threshold.

Given $\{C_k\}_{k=1}^{m}$, the feature weights W are randomly initialized and C and M can be updated by the equations given below in 4.11 & 4.12 respectively.

$$C = \{(X_k, Y_k) = \arg\min_{x \in C_k, y \in C_{-k}} d(X, Y, W) \}_{k=1}^{m} \quad (4.11)$$

$$M = \{(X_k^*, Y_k^*) = \arg\min_{x \in C_k^1, y \in C_k^2} d(X, Y, W) \}_{k=1}^{m} \quad (4.12)$$

where

- $C_{-k}$ is the set of points except $C_k$
- $C_k^1$ and $C_k^2$ are sub clusters of $C_k$

By the definition of single link clustering we have

$$d(X, Y, W) > \epsilon \text{ for all } (X, Y) \in C \quad (4.13)$$

$$d(X, Y, W) \leq \epsilon \text{ for all } (X, Y) \in M \quad (4.14)$$

C and M corresponds to positive and negative sample set, such that feature weights and threshold can be learned by minimizing the classification.
error. For this learning we use logistic regression and the objective function can be defined as in equation 4.15.

\[ J(\theta; C, M) = \frac{-1}{2m} \left\{ \sum_{(x,y) \in C} \log \left( h_\theta \left( X_{x,y}' \right) \right) + \sum_{(x,y) \in M} \log \left( 1 - h_\theta \left( X_{x,y}' \right) \right) \right\} \]  

(4.15)

where

\[ h_\theta \left( X_{x,y}' \right) = \frac{1}{\left( 1 + \exp \left( -\theta^T X_{x,y} \right) \right)} \]

\[ \theta = \left( \begin{array}{c} \epsilon \\ w \end{array} \right) ; \quad X_{x,y}' = \left( \begin{array}{c} 1 \\ X_{x,y} \end{array} \right) \]

By minimizing the objective function \( J(\theta; C, M) \) with respect to \( C \) and \( M \) the weights and the threshold can be estimated by the equation 4.16 given below.

\[ \theta^* = \arg \min_{\theta} J(\theta; C, M) \]  

(4.16)

For the above equation the required feature weights are greater than or equal to zero. Minimizing the objective function can create non-linear optimization problem and solved by classic gradient optimization methods by Hastie et al (2009).

In order to generate set \( C \) and \( M \), the initial value of \( W \) has been specified first. Here we are using iterative optimization which involves two steps for the computation of \( C \) and \( M \). This algorithm is called as self training distance metric algorithm. The pseudocode for self training distance metric learning algorithm is given below.

Algorithm:

Input: labeled clusters set \( \{C_k\}_{k=1}^m \)
Output: optimized \( \theta \) such that \( J \) is minimized
Procedure: initialize the value \( \theta \) randomly
Repeat
Stage 1: update C and M according to the equations, from current assignment of \( \theta \) feature weights \( W \) are computed.

Stage 2: Given positive samples C and negative samples M, use logistic regression to estimate the optimized parameter, update \( \theta \) until reach convergence.

Given, the top level cluster set \( \{C_k\}_{k=1}^m \), the self training distance metric learning algorithm finds an optimized \( \theta \) such that \( J(\theta;C,M) \) is minimized with respect to C and M. In self training distance metric algorithm initially one value for \( \theta \) is set before the iteration begins. In first stage according to equation 4.11 and 4.12, C and M are updated with respect to current assignment of \( \theta \). In second stage by minimizing the objective function with respect to current assignment C and M, \( \theta \) is updated. These two stages are repeated until maximum number of iteration is exceeded.

4.3.1.2.3 Empirical analysis

Usually by using this algorithms, it achieve good performance after a very small number of iterations. Here 466 texts in ICDAR 2011 training database are labelled and 70% of them as training data and 30% as validation data. Text in the same image corresponds to top top level cluster set in equation (4.11) and (4.12).

In stage 1, for different images C and M are computed separately but are merged to feed optimization process in next stage using L-BFGS method proposed by Nocedal (1980) and the parameters are updated. In second stage, the performance of land distance weights and threshold are evaluated in each iteration. It shows very low error rate for the first several iterations and no major improvement in continuing iterations.
We plot the objective function after the first stage and second stage in each iteration for converged one and not converged one as shown in Figure 4.3(a). The error rates on validation set are plotted as shown in figure 7(b).

![Objective function value](image)

(a) ![Validation set error rate](image)

(b)

**Figure 4.3 Objective function value**

From the above figure we notice that the objective function and the error rate drop immediately after several iterations.

### 4.3.1.3 Text elimination

Using the text construction algorithm only 9% of text candidates correspond to two texts are identified. So it is hard to find effective test in an unbalanced data based. Before applying text classification most of the non-text candidates should be removed. We use a character classifier to find the most occurrence probability of text candidates corresponding to non-text and it remove text candidates with high non-text probability.

Text region height, width and aspect ratio, smoothness and stroke width are the features used to train the character classifier. Smoothness is defined as the average difference of adjacent boundary pixels. Stroke width includes mean and variance of a character. Characters like ‘I’, ‘J’ and ‘L’ having small aspect ratio and it can be labelled as negative samples.
T be the text candidates, let $O(m, n; p)$ be the observation that there are $m$ ($m \in \mathbb{N}, m \geq 2$) character candidates in $T$, where $n$ ($n \in \mathbb{N}, n \leq m$) candidates are classified as non-characters by a character classifier of accuracy (on the validation set) $p$ ($0 < p < 1$). The probabilities of the observation conditioning on $T$ corresponding to text and non-text are $P(O(m, n; p) | \text{text}) = pm - n(1 - p)n$ and $P(O(m, n; p) | \text{non-text}) = (1 - p)m - npn$ respectively.

Let $P(\text{text})$ and $P(\text{non-text})$ be the probability of $T$ corresponding to text and non-text. By applying Bayes’ rule, the posterior probability of $T$ corresponding to non-text given the observation is shown in equation 4.17.

$$P(\text{non-text} | O(m, n; p)) = \frac{P(O(m, n; p) | \text{non-text})P(\text{non-text})}{P(O(m, n; p))} \tag{4.17}$$

where $P(O(m, n; p))$ is the probability of the observation

$$P(O(m, n; p)) = P(O(m, n; p) | \text{text})P(\text{text}) + P(O(m, n; p) | \text{non-text})P(\text{non-text})$$

The candidate region is rejected if

$$P(\text{non-text} | O(m, n; p)) \geq \varepsilon \tag{4.18}$$

where $\varepsilon$ is the threshold.

Text candidates of different lengths tend to have different probability of being text.
Given a text candidate $T$ having length $l$, let $N_l$ be the total number of text candidates of length $l$, $N^*_l$ be the number of text candidates of length $l$ that correspond to text, we estimate the prior of $T$ being text as $P_l(\text{text}) = \frac{N^*_l}{N_l}$, and the prior of $T$ being non-text as $P_l(\text{non-text}) = 1 - P_l(\text{text})$. For a length ‘L’ if no text candidates are found $P_L(\text{text})$ is set as 0.5 as a default value. The precision, recall and f-measure of text candidate classification are plotted as shown in Figure 4.4. When $\epsilon$ increases text candidates most unlikely to be eliminate so it increase the recall value.

For a text detection normally recall is preferred over precision until reached $\epsilon = 0.995$. So 92.95% text are preserved while 95.25% non text are eliminated as shown in Figure 4.5.
4.3.1.4 Candidates classification

Arbitrary orientation text detection from compound image is much more complicated than horizontal text detection. The difficulty is to regularize the text construction process due to text line alignment feature. In such cases text alignment feature is always there but they are in different orientation.

In clustering based text construction algorithm arbitrary orientation feature cannot be used. So we are using arbitrary orientation text detection from compound images using forward-backward algorithm. By using this forward-backward algorithm text lines are detected. After the orientation of text lines are computed and convert arbitrary orientation text line to horizontal text lines as shown in Figure 4.6(a). Then for the converted horizontal text line detection text construction, elimination and classification are also be performed consecutively.

Figure 4.6 Forward backward algorithm
After character extraction according to their priorities the pairs of components are sorted.

(char,char) is at highest priority
(non char,char) is at medium priority
(char,non char) is at medium priority
(non char,non char) is at lowest priority.

After sort the pair of components forward-backward algorithm begins with component pair(C1,C2) with highest priority and tries to expand towards forward direction from C1 to C2 as shown in Figure 4.6(b) and (c).

If the forward direction is no longer expandable it tries in backward direction. After backward direction is no longer expandable we have finished the text line detection as shown in Figure 4.6(d).

The component pairs containing components appearing in the detected text line can be erased from the component pair sequence.

Then it detect the next line by starting with next highest priority component pair. Component candidates searching and optimal component candidate selecting are the two steps involved in the component expansion process. The angle requirement is the angle between Euclidean vector and the Euclidean vector from C2 to C3 as shown in Figure (e) and (f). The Euclidean vector should not exceed the threshold.

After finding all the candidates optimal component is selected from candidates based on two criteria.ie) It should be close to current component and the line between centroids of current component and optimal candidate should not intersects with other candidates. In the forward-backward algorithm we still using text line alignment feature.
After the text can be extracted from the compound image a separate coding can be performed for the text block which is described in 4.3.2.

4.3.2 Text Block Coding

In text block compression, the text/graphics blocks are subjected to color quantization. Compression is performed on quantized blocks using the Huffman coding. Color Quantization algorithm and Huffman coding techniques are specified in the subsequent sections in detail.

4.3.2.1 Color quantization algorithm

The text blocks are typically dominated by several major colors. The colors with high frequency i.e. the colors having higher probability are chosen as major colors. In this work, only four colors are considered as major colors. If more than four colors are having higher probability, then the first four colors having the largest luminance value in the luminance histogram will be chosen as major colors. The term luminance represents the perceived brightness of a pixel.

The luminance value for each color is computed from RGB via the formula given in equation 4.19.

\[ \text{RGB Luminance value} = (0.3R) (0.59G) (0.11B) \]  \hspace{1cm} (4.19)

For instance let us consider an RGB color of (100,150, and 200) then its luminance value will be computed as 100 0.3 150 0.59 200 0.11 140

The colors whose values are closer to the major colors within the given distance threshold are quantized to their corresponding major colors
using the color quantization algorithm proposed by Wenpeng Ding (2006) which is described as follows.

**Algorithm 1: Color Quantization**

Find first four major colors $A_0, A_1, A_2, A_3$;

FOR each pixel $P_i$

FOR each major color $j$

IF 

$|P_i A_j| < Th$ for some $j$, THEN

$P_i : A_j$;

ENDIF;

ENDFOR;

ENDFOR;

Primary colors are selected based on the above mentioned algorithm and their corresponding color values are shown in Table 4.1.

**Table 4.1 Primary colors value**

<table>
<thead>
<tr>
<th>Color</th>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>200</td>
<td>154</td>
<td>121</td>
</tr>
<tr>
<td>C2</td>
<td>199</td>
<td>153</td>
<td>120</td>
</tr>
<tr>
<td>C3</td>
<td>202</td>
<td>156</td>
<td>123</td>
</tr>
<tr>
<td>C4</td>
<td>203</td>
<td>156</td>
<td>126</td>
</tr>
</tbody>
</table>

A Color index value is assigned to pixels of the text block. The index values 0, 1, 2 and 3 are assigned to the four major colors respectively. All the other colors are assigned with the index value 4. The major colors in each block are recorded.
The index value of the text block is scanned and a raster scanning order based compression is performed. The current pixel index value is based on its causal neighbors as shown in Figure 4.7.

<table>
<thead>
<tr>
<th>$N_1$</th>
<th>$N_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_3$</td>
<td>$P_x$</td>
</tr>
</tbody>
</table>

**Figure 4.7 Text block coding contexts**

where “$P_x$” is the current pixel that is to be coded. Each neighbour may have five different index values; for coding the current pixel $P_x$ there are a total of $5^3$ (125) contexts.

**Table 4.2 Index values for the text block pixels**

<table>
<thead>
<tr>
<th>4 4 4 4 4 4 4 4 4 4 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 4 4 4 4 4 4 4 4 3 1</td>
</tr>
<tr>
<td>4 3 3 3 1 3 4 4 3 4 4</td>
</tr>
<tr>
<td>3 1 1 1 1 1 4 4 2 1 4</td>
</tr>
<tr>
<td>2 2 2 2 1 3 4 1 1 3 1</td>
</tr>
<tr>
<td>2 2 2 2 1 3 3 1 3 1 1</td>
</tr>
<tr>
<td>2 1 1 1 2 1 3 2 1 2 2</td>
</tr>
<tr>
<td>1 1 1 2 2 1 1 2 1 1 2</td>
</tr>
<tr>
<td>1 2 2 2 2 2 2 1 3 3 1</td>
</tr>
<tr>
<td>3 2 2 2 2 2 1 1 1 1 1</td>
</tr>
<tr>
<td>2 2 2 2 2 2 2 2 2 2 2</td>
</tr>
<tr>
<td>4 4 4 4 4 4 4 4 4 4 4</td>
</tr>
</tbody>
</table>

The color index values for the text blocks are shown in Table 4.2. The index of the current pixel is coded by a simple Huffman coder using the
context specified by its three casual neighbours \([N_1, N_2, N_3]\). If the index of the current pixel is 4, then a Huffman coder is used for coding of that pixel.

After the quantization of color has been performed, the index value or map vector is assigned to each primary color. The map vector assigned to one of the text blocks is as follows. The text block coding algorithm are as follows:

**Algorithm 2: Text block coding**

Table 2 Index values for the text block pixels
Quantize the text block using Color Quantization;
Convert the block pixels to index values;
FOR row \(R = 1\) to last row
FOR column \(C = 1\) to last column
Get index value for the current pixel;
/*Generate the Context value for current pixel*/
Context : = \(N_1\ 25\ N_2\ 5\ N_3\)
Code current pixel index using the context;
IF pixel index is 4 (other colors) THEN
Code the current pixel value;
ENDIF
ENDFOR
ENDFOR

Table 4.3 Index and Probability

<table>
<thead>
<tr>
<th>Color index</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.26389</td>
</tr>
<tr>
<td>2</td>
<td>0.31944</td>
</tr>
<tr>
<td>3</td>
<td>0.125</td>
</tr>
<tr>
<td>4</td>
<td>0.29167</td>
</tr>
</tbody>
</table>
The color index and their corresponding probability values are given in Table 4.3. The color quantization algorithm is applied and the coding of text pixels are done based on their index value. The coding algorithm used here is a Huffman coding which is described in the following section.

4.3.2.2 Huffman coding

Mabberg et al (2015) proposed a generalized Huffman coding for binary trees with choosable edge lengths, in which a plasmid based library with efficient and reliable information retrieval and assembly with uniquely designed primers are described.

The Huffman code procedure is based on the following two methods.
a. Index values that occur more frequently i.e. the higher probability symbols will have shorter codewords than symbols that occur less frequently.
b. The same length is assigned for the two lowest probability symbols. In Huffman coding, the two lowest probability symbols are merged.

The probability of the index values or map vector 1,2,3,4 is calculated first. For instance, the probability of the index values in one of the text blocks is specified below.

This process is repeated until only two probabilities of two compound symbols are left. Then, a code tree is generated and Huffman codes are obtained from the labeling of the code tree. This is illustrated with an example which is shown in Table 4.4. Where I-Index, P-Probability.
Table 4.4 Huffman source reduction

<table>
<thead>
<tr>
<th>Input</th>
<th>Index</th>
<th>Reduced</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>P</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0.31944</td>
<td>0.38889</td>
<td>0.61111</td>
</tr>
<tr>
<td>4</td>
<td>0.29167</td>
<td>0.31944</td>
<td>0.38889</td>
</tr>
<tr>
<td>1</td>
<td>0.26389</td>
<td>0.29167</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.125</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

At the left side of Table 4.4, the index and their subsequent probabilities are arranged in descending order. The least probabilities 0.26389 and 0.125 are merged together; this provides a compound symbol with probability 0.38889. The compound symbol’s probability is placed in the index reduction column 1 and the probabilities are arranged in decreasing order. The process is continued until only two probabilities are left.

The next step is the coding of each reduced source. This is accomplished by coding the smallest source and working back to its original source. The symbols 0 and 1 are used for coding the two reduced index. As shown in Table 4.4, two indices on the right are assigned with these codes. As the reduced source index with probability 0.38889 is generated by combining the two indices in the reduced source to its left, the 0 used to code it is now assigned to both of these indices and they are distinguished from each other by appending a 0 and 1. For each reduced index, this operation is repeated until the original source index is reached. The resulting code is shown at the far-left in Table 4.2. The average length of the code is given by the average of the product of probability of the index and number of bits used to encode it. This is calculated as:

\[
L_{avg} = (0.31944) \cdot (2) + (0.29167) \cdot (2)+(0.26389) \cdot (2)+(0.125) \cdot (2)
\]

= 2 bits/symbol.
In this way, Huffman’s procedure creates the optimal code for a set of indices and probabilities subject to the constraint that the indices be coded one at a time.

4.3.2.3 Huffman decoding

After the code assignment process is done, coding and/or decoding is accomplished in a simple lookup table. The code itself is an instantaneous uniquely decodable block code. Since each source symbol is mapped into a fixed sequence of code symbols it is called a block code. Each code word in the string of the code indices are decoded without referencing succeeding indices. Since any string of code indices can be decoded in only one way. By examining the individual indices of the resulting string in a left to right manner, any string of Huffman encoded indices can be decoded. Table 4.5 shows the Huffman code assignment procedure for each index value.

<table>
<thead>
<tr>
<th>Index</th>
<th>2</th>
<th>4</th>
<th>1</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prob</td>
<td>0.31944</td>
<td>0.29167</td>
<td>0.26389</td>
<td>0.125</td>
</tr>
<tr>
<td>Avg</td>
<td>0.63888</td>
<td>0.58334</td>
<td>0.52778</td>
<td>0.250</td>
</tr>
</tbody>
</table>

For the binary code of table, a left-to-right scan of the encoded string 10001101 reveals that the first valid code word is 10, which is the code for index value 1. The next valid code 00, represents the index value 2. The valid code for the index 3 is 11. The valid code for the index 4 is 01. Thus, the completely decoded index values 1,2,3,4 are revealed by this process. Making use of the Huffman decoding, the original image or data can be decompressed as explained above. First, a probability distribution is performed and a code table is made for the coding and the decoding process. This implementation presents a recursion process for the coding and decoding process.
4.3.2.4 Cost reduction of Huffman table

In the modified Huffman coding system the cost of the Huffman code table is reduced. The first part of this technique is to divide the highest two probabilities from the image and allocate 0 and 1 to those probabilities and for the remaining probabilities do binary Huffman coding in the second part. By performing this, we can send the highest probability pixel by using a single bit itself. For a typical binary Huffman coding, more than one bit code for the highest probability pixel may occur. Hence, sending more bits with higher probability at many times is wastage of memory. The reduced Huffman cost table obtained from Table 4.5 is shown in Table 4.6.

<table>
<thead>
<tr>
<th>I</th>
<th>P</th>
<th>Code</th>
<th>I</th>
<th>Code</th>
<th>2</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.31944</td>
<td>00</td>
<td>0.38889</td>
<td>1</td>
<td>0.61111</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0.29167</td>
<td>01</td>
<td>0.31944</td>
<td>00</td>
<td>0.38889</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.26389</td>
<td>10</td>
<td>0.29167</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.125</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Our above mentioned procedure of Huffman Coding for the compression and decompression of text/graphics block results in a better compressed and decompressed text/graphics block.

4.3.3 Picture Block Coding

The pictorial blocks are compressed using SVD based compression technique. The decomposition technique has been widely used in signal processing and statistics. There are many methods to decompose a matrix into more useful elements. One of the most famous factorization is singular value decomposition (SVD). This SVD can be applied to rectangular a matrix which contains both real and complex. It is one of the most useful tools of linear
algebra. It depends upon two techniques namely factorization and approximation. The SVD can work wonderfully with both under-determined matrix and over-determined matrix.

Let A be the mxn matrix of real-valued or complex-valued data with rank r. The maximal number of linearly independent rows or columns of A gives rank r. The real-valued matrix can be represented as shown in equation 4.20.

\[ A_{mxn} = U_{mxm} S_{mxn} V^T_{nxn} \]  
(4.20)

U is the mxn orthogonal matrix.

\[ U^T U = I_{mxm} \]  
(4.21)

I be the mxm identity matrix.

S be the mxn diagonal matrix with non-negative real numbers.

V^T be the transposed matrix of the nxn orthogonal matrix V.

This factorization is called as singular-value decomposition of A. In simple, the matrix A can be decomposed as a product of three matrices as in equation 4.20.

S can be ordered in a non-increasing way and it is uniquely determined by A. The diagonal entries of S are known as singular values of A. Whereas, the matrices U and V are not uniquely determined by A. The columns of U forms a orthogonal ‘output’ basis vector called left singular vectors and rows of V^T forms a set of orthogonal ‘input’ basis vector called right singular vectors.

4.3.3.1 SVD in image compression

SVD is a very powerful technique which deals with set of equations or matrices. For a rectangular matrix SVD is an important factorization
method, which can be applied in image compression. It has been used for several applications in signal processing and statistics.

To compress an image using SVD, several steps should be followed carefully. Initially set the pixel image as mxn matrix. An mxn image is an m x n matrix and the entry \((i,j)\) is interpreted with brightness of pixel \((i,j)\). That means the matrix entries can be interpreted as pixels ranging from black(0) through several shades of gray to white(1). This can also present a colorful image.

Let \(A = USV^T\) be the SVD of \(A\). We write
\[
U = [u_1, u_2, \ldots, u_m], \quad \text{and} \quad V = [v_1, v_2, \ldots, v_n].
\]

Matrix \(A\) can be written as
\[
A = USV^T = \sum_{i=1}^{r} \sigma_i U_i V_i^T.
\]

As \(\sigma_j = 0\) for \(j > r\) where \(r\) is the rank of the matrix \(A\),
\[
A = \sum_{i=1}^{r} \sigma_i U_i V_i^T.
\]

Rank-k approximation of matrix \(A\) can be written as
\[
A_k = \sum_{i=1}^{k} \sigma_i U_i V_i^T.
\]

The best approximation in the sense of minimizing the \(L_2\)-norm of the error
\[
\|A - A\|_2 = \sigma_{k+1}.
\]

Also \(A_k\) can be expressed as
\[
A_k = US_k V^T
\]

where \(S_k = \text{diag}(\lambda_1, \ldots, \lambda_k, 0, \ldots, 0)\).
The length of a vector \( x = (x_1, x_2, \ldots, x_n)^T \) is usually given by the Euclidean norm
\[
||x||_2 = (x_1^2 + x_2^2 + \ldots + x_n^2)^{1/2}
\] (4.27)

In the case of the Euclidean norm and square matrices, the induced matrix norm is considered as spectral norm. The largest singular value of \( A \) is the spectral norm of \( A \) or it is the square root of the largest eigenvalue of the positive-semidefinite matrix \( A^*A \).

\[
||A||_2 = \sqrt{\lambda_{\text{max}}(A^*A)}
\] (4.28)

where \( A^* \) denotes the conjugate transpose of \( A \). \( A_k \) has rank \( k \) and it can be represented as
\[
A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T
\] (4.29)
\[
= U \begin{bmatrix}
\sigma_1 & \sigma_2 & \ldots & \ldots \\
\ldots & \sigma_2 & \ldots & \ldots \\
\ldots & \ldots & \ldots & \sigma_k \\
\end{bmatrix} V^T
\] (4.30)

The L2 norm of the error can be represented as
\[
||A - A_k||_2 = ||\sum_{i=k+1}^{n} \sigma_i u_i v_i^T||_2
\]
\[
= \sigma_{k+1}
\] (4.31)

\( m \cdot k + n \cdot k = (m + n) \cdot k \) memory places to store \( u_1 \) through \( u_k \) and \( \lambda_1 v_1 \) through \( \lambda_k v_k \). It can be used to reconstruct the image \( A_k \) or the matrix \( A_k \). When comparing with the storage places, the place needed to store the decomposed matrix is less than the place needed to store original matrix \( A \). \( A_k \) is the compressed image, using \( (m + n) \cdot k \) memory places. By altering the values of \( k \) different errors and compression degrees are occurred.
The different errors are \( \|A - Ak\|^2 /\|A\|^2 \) and compression degrees is defined as \( 1 - (m + n) \cdot k/(m \cdot n) \).

It is very clear that the matrix \( A_k \) contains less information than the original matrix \( A \). While, considering the requirements of human visual, selecting the suitable value \( k<r \) for the image file \( A_k \), getting a good approximation of a form \( A_k \). When \( K \) is the smaller value, less data to present \( A_k \). Where \( k \) gets closer to rank ‘r’, then \( A_k \) will approach the original image matrix \( A \). By selecting the appropriate number of singular values, the compressed matrix \( A_k \) will be close to the original matrix.

Usually, for \( A_{mxn} \), with \( 256 \leq n \leq 2048 \), it attain a good quality image by choosing the \( k \) values between 25 and 100. Then for a square matrix \( m=n \) and \( r=n \), if the range of \( k \) is between \( r/5 \) to \( r/30 \), then the compression ratio will be between \( 3/5 \) to \( 14/15 \).

The commands used for SVD was given below:

```
load A.mat;
[U, S, V]=svd(X);
colormap('gray');
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k'));
```

Initially it load the image in MATLAB as a matrix \( A \). Then SVD function is used to decompose the matrix into U,S and V. Colormap is an \( m \)-by-\( 3 \) matrix of real numbers between 0.0 and 1.0. Each row is an RGB (red, green, blue) vector that defines one color.

\( A = \) randint(8, 10, \([-64, 64]\)) is a MATLAB command which create the matrix of random integers.

The function randint \((m,n,rg)\) which use here generates an \( 8 \)-by-\( 10 \) \((m\text{-by-}n)\) integer matrix with element in the range \([-64, 64]\) \(rg\).
Now SVD can be achieved by $[U, S, V] = \text{SVD}(A)$ and the rank of the matrix $A$ is achieved by using the MATLAB command $\text{rank}(A)$ or by $\text{diag}(S)$.

For example, the image matrix $A_{9 \times 10}$ is decomposed by SVD into three matrices: $U(9 \times 9)$, $S(9 \times 10)$, $V (10 \times 10)$.

$$A = \begin{bmatrix}
68 & 71 & 63 & 63 & 61 & 64 & 60 & 67 & 66 & 63 \\
67 & 64 & 64 & 61 & 63 & 65 & 66 & 77 & 70 & 66 \\
69 & 63 & 64 & 63 & 69 & 194 & 201 & 197 & 193 & 92 \\
67 & 67 & 65 & 65 & 81 & 112 & 54 & 87 & 85 & 147 \\
66 & 68 & 68 & 72 & 59 & 90 & 57 & 54 & 84 & 139 \\
67 & 61 & 70 & 75 & 83 & 90 & 96 & 101 & 107 & 64 \\
68 & 72 & 77 & 68 & 84 & 92 & 100 & 101 & 70 & 145 \\
65 & 65 & 62 & 72 & 84 & 93 & 104 & 130 & 101 & 134 \\
65 & 61 & 62 & 69 & 81 & 88 & 123 & 113 & 105 & 122
\end{bmatrix}$$

$$U = \begin{bmatrix}
-.239 & -.146 & .519 & .147 & -.050 & -.297 & -.344 & .306 & .573 \\
-.248 & -.103 & .457 & .031 & .062 & -.197 & -.255 & -.039 & -.780 \\
-.489 & .781 & .218 & .240 & -.058 & -.196 & -.062 & -.024 & .000 \\
-.320 & -.344 & -.331 & -.395 & .582 & .007 & .175 & .373 & -.063 \\
-.288 & -.386 & -.202 & .656 & -.593 & .229 & -.128 & -.297 & -.005 \\
-.310 & .085 & .522 & .066 & .140 & .417 & .581 & -.271 & .124 \\
-.337 & -.277 & -.159 & .394 & -.114 & -.628 & .393 & -.252 & .069 \\
-.356 & -.083 & -.149 & -.420 & .367 & .309 & -.517 & -.380 & .159 \\
-.348 & -.009 & -.060 & -.458 & -.363 & .344 & .080 & .629 & -.115
\end{bmatrix}$$

$$S = \begin{bmatrix}
833.208 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 164.686 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 76.291 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 55.314 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 29.909 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 25.309 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 16.026 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6.704 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3.756 & 0
\end{bmatrix}$$
If \( k = 4 \), then we get the \( U_k, S_k, V_k \) as

\[
U_k = \begin{pmatrix}
-0.239 & -0.146 & 0.519 & 0.147 \\
-0.248 & -0.103 & 0.457 & 0.031 \\
-0.489 & 0.781 & -0.218 & 0.240 \\
-0.320 & -0.344 & -0.331 & 0.395 \\
\end{pmatrix}
\]

(4.36)

\[
S_k = \begin{pmatrix}
833.208 & 0 & 0 & 0 \\
0 & 164.6863 & 0 & 0 \\
0 & 0 & 76.291 & 0 \\
0 & 0 & 0 & 55.3135 \\
\end{pmatrix}
\]

(4.37)
\[
V_k = \begin{pmatrix} 
-0.236 & -0.186 & 0.339 & 0.117 \\
-0.231 & -0.230 & 0.307 & 0.112 \\
-0.233 & -0.216 & 0.314 & 0.070 \\
-0.238 & -0.216 & 0.317 & 0.034 \\
-0.263 & -0.219 & 0.251 & -0.235 \\
-0.373 & 0.217 & -0.283 & 0.625 \\
-0.367 & 0.434 & -0.010 & -0.471 \\
-0.389 & 0.314 & -0.022 & -0.381 \\
-0.370 & 0.319 & 0.030 & 0.361 \\
-0.387 & -0.579 & -0.670 & -0.158 
\end{pmatrix} 
\]

Calculating \( A_k = U_k S_k V_k^T \) now gives

\[
\begin{pmatrix} 
65.707 & 64.423 & 64.463 & 65.310 & 65.548 & 62.788 & 58.344 & 65.824 & 69.962 & 63.063 \\
63.812 & 62.367 & 62.750 & 63.858 & 66.289 & 64.469 & 67.238 & 73.526 & 72.457 & 66.073 \\
68.013 & 60.785 & 62.694 & 64.343 & 71.513 & 192.761 & 199.242 & 194.181 & 195.863 & 192.121 \\
67.463 & 69.248 & 67.907 & 68.478 & 71.068 & 107.979 & 63.281 & 78.233 & 87.714 & 49.529 \\
72.198 & 68.973 & 69.844 & 71.171 & 73.947 & 190.277 & 98.761 & 102.587 & 102.403 & 64.588 \\
68.092 & 69.134 & 69.897 & 72.151 & 85.919 & 84.692 & 93.782 & 103.648 & 81.175 & 146.667 \\
65.885 & 65.446 & 66.744 & 69.134 & 83.513 & 96.270 & 113.956 & 120.204 & 69.547 & 133.947 \\
63.996 & 62.855 & 64.446 & 66.889 & 81.163 & 93.056 & 117.597 & 121.943 & 97.289 & 119.852 
\end{pmatrix} 
\]

Comparing \( A \) and \( A_K \), \( A_k \approx A \). For a larger matrix select the value of ‘k’ as small, so it gives high compression ratio. By using this SVD technique the original image can be compressed to 80-90% of the original one.

In MATLAB, each entry in the matrix corresponds to a small
square of image. The value of each entry corresponds to a color. In MATLAB, the color spectrum can be obtained by using the command `image(A);`

![Color spectrum and blocked image](image.png)

**Figure 4.8 Color spectrum and blocked image**

In Figure 4.8 contains nine square blocks forms one large block. The code used is, `A=randint(3,3); image(A);`

By previous theory, A can be approximated using smaller number of iterations. Here three iteration images with three different parameters of $\sigma_k, \sigma_1, \sigma_2, \sigma_3$ are performed and the number of iterations equals the rank of the approximate SVD matrix $A_k$. The original image occur at the third iteration as shown in Figure 4.9.

![Iteration images](iteration.png)

**Figure 4.9 SVD compression-original image at third iteration**

Consider if A be 15x20 matrix of integers ranging from -64 to 64.
With rank 12, the original image can be represented in the twelfth iteration. The command used in matlab is, \( A = \text{radiant}(15, 20, 64); \) \([U, S, V] = \text{Svd}(A);\)

And the approximation \( A_k = U_k S_k V_k^T \) for \( k \) iteration or random matrix with \( r=12 \) we get a good quality image at the tenth iteration itself as shown in Figure 4.10.

![Original image](image1.jpg) ![Five iterations](image2.jpg) ![Twelve iterations](image3.jpg)

**Figure 4.10 SVD compression-15x20 matrix of random integers**

A 512x512 natural image is performed by the SVD technique, At 10\(^{th}\) iteration we identify that what the image is as shown in Figure 4.11, At 25\(^{th}\) iteration the image is much clear and at 75\(^{th}\) iteration image is mostly same as the original image.
Figure 4.11 SVD based compression for 512x512 real image

The compression degree for this image is given by the equation given below:
Compression degree  =  1 − \frac{(512+512) \times 75}{512 \times 512} \tag{4.40}
= 71 \%

The difference between the compressed and the original image is calculated by

\[ L_2\text{-norm, } \|A-A_k\|_2 = \sigma_{k+1} \] \tag{4.41}

For a color image, at first the image is divided into three layer (red, green and blue) for each layer decomposition and compression is performed separately. Finally all the three compressed matrixes are combined together to get original image.

4.3.4 Encryption Algorithm

Then the compressed blocks (text and image compressed blocks) are merged together and the encryption algorithm has been performed.

In the encryption algorithm DWT computes the approximation coefficient matrixes \( LL_1 \) and detailed coefficient matrices, \( LH_1, HL_1 \text{ & } HH_1 \). Second wavelet decomposition resulting in \( LL_2, LH_2, HL_2 \text{ & } HH_2 \) are formed by the approximation matrix \( LL_1 \). The encryption algorithm starts from the second wavelet decomposition & eliminates \( LL_2 \) subband values. It is important that set the values in \( LL_2 \) subband so that it will indistinguishable from the other subband values. This can be done by using a weighting factor as in equation 4.42.

\[ \text{The weighting factor is given by } LL_2(i,j) = LL_2(i,j)/(mn) \] \tag{4.42}

where \( m \) and \( n \) are the matrix dimension of \( LL_2 \).
This can be used as encryption key in this algorithm. The second step is reserving the sign of other frequencies. It can be done by multiplying the detailed coefficient matrices with 1 as shown below in equation 4.43.

\[
\begin{align*}
\text{LH}_2 &= \text{LH}_2 \times 1, \\
\text{HL}_2 &= \text{HL}_2 \times 1, \\
\text{HH}_2 &= \text{HH}_2 \times 1.
\end{align*}
\] (4.43)

The purpose of sign reverse operation is that the magnitude of the sinusoid corresponds to its contrast. A negative magnitude represents contrast reversal (i.e., dark become bright & bright become dark).

And the third step is swapping the contents of \( \text{HL}_2 \) matrix with \( \text{HH}_2 \) and swapping the contents of \( \text{HL}_2 \) matrix with \( \text{LL}_2 \) matrix. Due to high spatial frequency, sudden spatial changes can be occurring in the image. And due to low spatial frequency represent orientation & proportions. For this reason swapping the contents of four sub bands is to change low & high frequencies before inverse transform is applied.

In the swapping process \( \text{LL} \) contains lowest frequency in vertical & horizontal directions. These frequencies are essential to reconstruct the image. \( \text{HL} \) contains high frequencies in vertical directions and having residual vertical frequencies, \( \text{LH} \) contains high frequencies in horizontal directions and having residual horizontal frequencies, \( \text{HH} \) contains high frequencies in vertical & horizontal directions. All the four subband images contain all the information present in the original image. Next the inverse discrete wavelet transform will be applied to the image.

This process will be repeated again on the first level of wavelet decomposition. In this second iteration eliminate \( \text{LL}_1 \) values using same weighting factor.

\[
\text{LL}_1(\ i,j) = \text{LL}_1(i,j)/(m \times n)
\] (4.44)
The second step involves reversing the sign by multiplying a value of 1 with detailed coefficient matrices. Next swapping the contents of LH\(_1\) with HH\(_1\) & HL\(_1\) with LL\(_1\) matrices. Next the second level inverse discrete wavelet transform is performed to an image. The detailed block diagram of this encryption scheme is shown in Figure 4.12.

**Figure 4.12 Encryption algorithm**

In this encryption algorithm, it uses only two levels of wavelet decomposition to encrypt an image. When comparing with the N levels of wavelet decompositions, low number of computations are required.

### 4.3.5 Decryption Algorithm

The encryption algorithm uses the discrete wavelet transform to compute approximation coefficient matrix LL\(_1\) and detailed coefficient matrices of the original image using Haar transform. In the first wavelet decomposition LL\(_1\) is used to produce second level decomposition resulting in LL\(_2\), LH\(_2\), HL\(_2\) & HH\(_2\).
The first level of decryption starts at the second level decomposition. Swapping the contents of LH\textsubscript{2} with HH\textsubscript{2} & HL\textsubscript{2} with LL\textsubscript{2} should be performed. Next reverse the sign by multiplying the value of 1 with detailed coefficient matrices.

\[ LH_2 = LH_2 \times 1, \quad HL_2 = HL_2 \times 1, \quad HH_2 = HH_2 \times 1. \quad (4.45) \]

Next reveal the LL\textsubscript{2} values using the same waiting factor as LL\textsubscript{2}(i,j) = LL\textsubscript{2}(i,j)x(mx\textsubscript{n}). The last stage of decoding the second level decomposition involves inverse wavelet transformation to retrieve the pixel values.

The process is repeated for first wavelet decomposition. Initially swapping the contents LH\textsubscript{1} with HH\textsubscript{1} and HL\textsubscript{1} with LL\textsubscript{1} should be performed. The next stage involves reversing the sign.
\[ LH_1 = LH_1 \times 1, HL_1 = HL_1 \times 1, HH_1 = HH_1 \times 1 \quad (4.46) \]

The next stage involves revealing the \( LL_1 \) value using the same weighting factor as given below in equation 4.47.

\[ LL_1(i,j) = LL_1(i,j) \times (m \times n) \quad (4.47) \]

Finally, inverse discrete wavelet transform is performed to obtain the original image.

### 4.4 EXPERIMENTAL RESULTS

The proposed methodology is implemented using MATLAB 7.10 and evaluated by testing the proposed scheme with an input compound image. The text and picture blocks are separated by MSER method and compression is applied for each text and picture block. The compressed image is then provided as an input to the encryption algorithm which converts the image value into an encrypted format. The original image, Encrypted, and the decompressed image is shown in Figure 4.14.

In the decryption stage, the image can be extracted using the decryption algorithm. Finally, the original image is reconstructed after the decompression process. Thus, a fast and secure image transmission can be achieved.

#### 4.4.1 Performance metrics

The proposed method achieves better compression ratio, memory size, compression time, decompression time and compared with existing compression techniques in terms of the above mentioned performance metrics.
4.4.1.1 **Compression ratio**

The compression ratio is the degree of data reduction obtained as a result of the compression process. This measures the ratio of the quantity of the compressed data in comparison to the quantity of the original data.

Compression Ratio = \( \frac{\text{Original size} - \text{compressed size}}{\text{original size}} \times 100 \)

![Original image](a)

![Encrypted image](b)

![Decompressed image](c)

**Figure 4.14 Original, encrypted and decompressed images**

4.4.1.2 **Compression and decompression time**

Compression and decompression time are the basic measurements used to estimate an image compression algorithm. Compression and decompression time signify the time taken for the algorithm to achieve the encoding and decoding processes correspondingly. Peak Signal to Noise Ratio is often used as a quality measurement between the original and compressed
image. A high PSNR indicates a better quality of the compressed or reconstructed image.

Table 4.7 Evaluation based on compression ratio

<table>
<thead>
<tr>
<th>Input</th>
<th>BFC</th>
<th>JPEG 2000</th>
<th>DjVu</th>
<th>XMLCC</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp 1</td>
<td>21.6</td>
<td>39.45</td>
<td>40.25</td>
<td>46.16</td>
<td>46.21</td>
</tr>
<tr>
<td>Comp 2</td>
<td>8.3</td>
<td>39.99</td>
<td>42.13</td>
<td>47.03</td>
<td>47.31</td>
</tr>
<tr>
<td>Comp 3</td>
<td>12.6</td>
<td>40.89</td>
<td>45.55</td>
<td>46.75</td>
<td>47.01</td>
</tr>
<tr>
<td>Comp 4</td>
<td>10.1</td>
<td>41.45</td>
<td>43.91</td>
<td>45.01</td>
<td>45.33</td>
</tr>
</tbody>
</table>

Table 4.8 Comparison of original image size/compressed image size (in kilobytes)

<table>
<thead>
<tr>
<th>SPEC</th>
<th>JPEG</th>
<th>LZW</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1255/65</td>
<td>112/6</td>
<td>98/63</td>
<td>225/36</td>
</tr>
<tr>
<td>1229/64</td>
<td>131/8</td>
<td>100/65</td>
<td>187/21</td>
</tr>
<tr>
<td>1009/82</td>
<td>210/15</td>
<td>131/83</td>
<td>156/17</td>
</tr>
<tr>
<td>717/47</td>
<td>685/32</td>
<td>73/45</td>
<td>456/52</td>
</tr>
</tbody>
</table>

Table 4.9 Comparison results of compression/decompression time

<table>
<thead>
<tr>
<th>Input</th>
<th>SPEC</th>
<th>JPEG</th>
<th>LZW</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp 1</td>
<td>124/29</td>
<td>124/71</td>
<td>223/23</td>
<td>121/25</td>
</tr>
<tr>
<td>Comp 2</td>
<td>128/31</td>
<td>122/70</td>
<td>225/29</td>
<td>117/22</td>
</tr>
<tr>
<td>Comp 3</td>
<td>162/42</td>
<td>111/64</td>
<td>224/28</td>
<td>108/20</td>
</tr>
<tr>
<td>Comp 4</td>
<td>172/75</td>
<td>117/67</td>
<td>282/39</td>
<td>104/18</td>
</tr>
</tbody>
</table>
Table 4.10 Evaluation based on PSNR values

<table>
<thead>
<tr>
<th>Input</th>
<th>DjVu</th>
<th>XMLCC</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp 1</td>
<td>44.66</td>
<td>46.16</td>
<td>47.66</td>
</tr>
<tr>
<td>Comp 2</td>
<td>46.33</td>
<td>47.03</td>
<td>48.53</td>
</tr>
<tr>
<td>Comp 3</td>
<td>45.92</td>
<td>46.75</td>
<td>46.89</td>
</tr>
<tr>
<td>Comp 4</td>
<td>43.92</td>
<td>45.01</td>
<td>46.12</td>
</tr>
</tbody>
</table>

The comparison results of this proposed work for compression ratio, memory size, compression time and decompression time are tabulated in the Tables 4.7–4.10 The evaluation of compression ratios and PSNR values of the proposed system and their comparisons with other techniques are illustrated in Figure 4.15 and 4.16 respectively.

Figure 4.15 Performance comparison based on compression ratio
Since a high value of the compression ratio and PSNR implies a better quality of the compressed image, it is obvious that the proposed method achieves better results.

4.5 DISCUSSION

This work has presented a novel block based compound image compression algorithm that enables an efficient compression of both text and pictorial blocks. The main contribution of this work is the development of an appropriate algorithm for separating the text/graphics and the picture blocks. A hybrid lossless coding method was designed for the compression of text/graphics block and a lossy coding method was developed for picture compression. Finally, an efficient encryption algorithm was designed for encryption of compressed compound images. Experimental results demonstrated that our novel approach has several benefits such as low complexity, excellent visual quality, competitive compression ratio, high PSNR value and most importantly fast and highly secure.
4.5 SUMMARY

Implementation of block based segmentation algorithm have been implemented to split text and pictorial blocks in the compound images and the compression results have been analyzed in this chapter. Further the complete research conclusion and future research scope are discussed in next chapter.