



FastSS: Fast and smooth segmentation of JPEG compressed printed text documents using DC and AC signal analysis

Bulla Rajesh¹ · Mohammed Javed¹ · P. Nagabhushan¹

Received: 30 November 2020 / Revised: 8 September 2021 / Accepted: 23 December 2021 /
Published online: 18 January 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

With the surge of COVID-19 pandemic, the world is moving towards digitization and automation more than it was presumed. The Internet is becoming one of the popular mediums for communication, and multimedia (image, audio, and video) combined with data compression techniques play a pivotal role in handling a huge volume of data that is being generated on a daily basis. Developing novel algorithms for automatic analysis of compressed data without decompression is the need of the present hour. JPEG is a popular compression algorithm supported in the digital electronics world that achieves compression by dividing the whole image into non-overlapping blocks of 8×8 pixels, and subsequently transforming each block using Discrete Cosine Transform (DCT). This research paper proposes to carry out Fast and Smooth Segmentation (FastSS) directly in JPEG compressed printed text document images at text-line and word-level using DC and AC signals. From each 8×8 block, DC and AC signals are analyzed for accomplishing Fast and Smooth segmentation, and subsequently, two Faster segmentation (MFastSS) algorithms are also devised using low resolution-images generated by mapping the DC signal (DC Reduced Image) and encoded DCT (ECM Image) coefficients separately. Proposed models are tested on various JPEG compressed printed text document images created with varied space and fonts. The experimental results have demonstrated that the direct analysis of compressed streams is computationally efficient, and has achieved speed gain more than 90% when compared to uncompressed domains.

Keywords Document Image processing · JPEG compressed domain · Printed text-line segmentation · Printed word segmentation · DCT coefficients

✉ Mohammed Javed
javed@iitita.ac.in

1 Introduction

Worldwide spread of COVID-19 has brought many changes in human life in order to avoid person to person physical or surface contact. In such a scenario automation and digitization is gaining much more importance in order to avoid any sort of human infection. Internet and digital technologies are being used as frequent mediums of communication. Due to the huge volume of data that is being generated, various multimedia (image, audio and video) tools combine data compression techniques for efficiency of storage and transfer. Automatic and direct analysis of compressed data is becoming an efficient and trending approach in handling and processing of such a large data, and this research field is termed as compressed domain processing in the literature [18, 26]. JPEG is one of the widely used digital image compression formats, and one sixth of the image data on the internet are being transferred in JPEG compressed file format [14, 31]. It has become the default file format for many digital tools, and due to which large volumes of printed text documents are being compressed with this file format [4, 10, 31, 45]. Therefore, this research paper aims at developing novel algorithms for direct analysis of compressed data in the JPEG compressed domain.

Analysis of text-line and word boundaries in printed text documents is a very significant problem in Document Image Analysis (DIA) [2, 18]. It is a crucial preprocessing stage for many important applications like Word Spotting, Document Retrieval and OCR [18, 25]. Although the problem of locating text boundaries in printed texts amidst varying font size and spacing looks to be an easy task. However, the same task becomes much more challenging when compressed documents are considered (see Sections 2 and 3). In the literature there are some attempts for the segmentation of text-lines and words in the pixel domain [1, 18, 22, 27, 30]. Global projection profile techniques have been used to extract the text-line boundaries [42]. These techniques are challenged with documents containing skew and tilt. To overcome such challenges piecewise projection profile techniques have been addressed in [27]. Subsequently Connected Component based analysis is reported in [22]. Painting based approaches have been explored in [1], where they smear the foreground portions of the image, and apply dilation operation on those foreground regions to obtain the single component for each text-line.

Text-line segmentation based on Hough transform on the subset of the document image is reported in [24]. Text-line segmentation by marking the central path within the area between two adjacent lines is discussed in [9]. Arabic handwritten text-line segmentation is proposed in [5]. Globally matched wavelet filters are used for segmentation of textual areas in document images [23]. Apart from conventional approaches suggested above, there have been attempts using Convolutional Neural Network (CNN) based approaches for the segmentation of text-lines and words [7, 30, 40]. Text-lines are identified by segmenting the baselines of each text-line which is reported in [30]. Another approach is discussed in [40], where text-lines have been marked in a handwritten document based on predicting the dense regions. Very recently a holistic approach for word recognition based on using CNN is addressed in [7]. All the segmentation techniques discussed above are generally meant to work with uncompressed/decompressed images, however, this research work proposes to directly analyze the JPEG compressed stream for the extraction of text-line and word segments in printed text documents.

In the literature, compressed streams of various file formats have been analyzed, and different segmentation techniques are reported directly on the compressed streams [18]. Some of them are like Run length [2, 18], JPEG [8, 19, 35], JBIG [28, 39], JPEG2000 [6, 15] and Fast H.264 [44]. Run length compressed domain approaches for text-line, Word and character

segmentation are reported in [16, 17]. This algorithm is applied directly on the run length compressed representation. Tunneling algorithm for tracing the text-line boundaries is proposed in [2]. Document segmentation using block activity is reported in JPEG compressed domain [8]. Here they made an auxiliary image using Encoding Cost Map (ECM) to segment different semantic regions in the image. Similarly, many transform coding techniques with lowest coding memory are discussed. For example, in [21, 37, 38] memory efficient and fast image coding for wavelet transform are explained. Automatic caption localization in JPEG compressed representation is reported in [45].

Detection of edges by performing the convolution operation directly on the compressed image is addressed in [33]. Low level feature extraction from the compressed image is reported in [34], where the edge information is estimated based on analyzing the AC coefficients values and signs. A recent work for marking the text-line and word segments in the JPEG domain is reported in [29]. Compressed image restoration is discussed in [36]. Steganalysis directly in the JPEG domain is reported in [43]. However, apart from these conventional approaches, different deep learning based models are also proposed to process the compressed data [13]. Here, instead of feeding pixel values, the compressed stream (DCT coefficients) is extracted from decompressing the compressed images, and fed to the convolutional neural network. However, this research paper proposes to explore the JPEG compressed stream for Fast and Smooth extraction of text-line and word boundaries in the printed text document images. To the best of our knowledge, such a work has not been reported in the literature.

This research paper investigates novel algorithms for Fast and Smooth segmentation of printed text-lines and words using DC and AC signals. DCT stream is extracted after applying the partial decompression on the JPEG compressed printed text document images. Approximate text-line and word boundaries are marked using the projection profile of the DC signal from each 8×8 DCT block. With the help of these approximate boundaries and AC signals different faster and smooth approaches are devised to extract text-line and word boundaries. All proposed approaches in this paper are categorized into four different techniques as shown in Fig. 1 (green color). First approach is Fast and Smooth Segmentation (FastSS-1) based on partial decompression of selected 8×8 DCT blocks. Similarly the second approach is Fast and Smooth Segmentation (FastSS-2) based on analyzing AC coefficients in selected 8×8 DCT blocks and without any decompression. Further, two Faster algorithms are proposed for More Faster and Smooth Segmentation (MFastSS-1 and MFastSS-2) in the JPEG compressed domain using the idea of extracting low resolution images from compressed data by [8, 29]. Low resolution images are generated by mapping the DC signal (called as DC Reduced image) and encoded DCT coefficients (called as ECM image). Due to this double compressed representation of JPEG compressed file, the proposed approaches are faster. This paper is specifically focused on JPEG compressed printed text document images, however to show the generality of the proposed segmentation algorithms, their working is also demonstrated on compressed handwritten text document images (as shown in Section 5).

Moreover, the proposed work provides further scope for many applications such as Word Spotting, Document Retrieval [11, 12, 17, 46], Optical Character Recognition (OCR) [20] and so on, directly in the JPEG compressed domain. Rest of the paper is organized as follows: Section 2 gives brief illustration of JPEG algorithm and DCT encoding on a printed text, Section 3 gives problem description with suitable illustrations, Section 4 discusses the proposed model and other relevant approaches, Section 5 reports the experimental results, Section 6 concludes with brief summary of entire work.

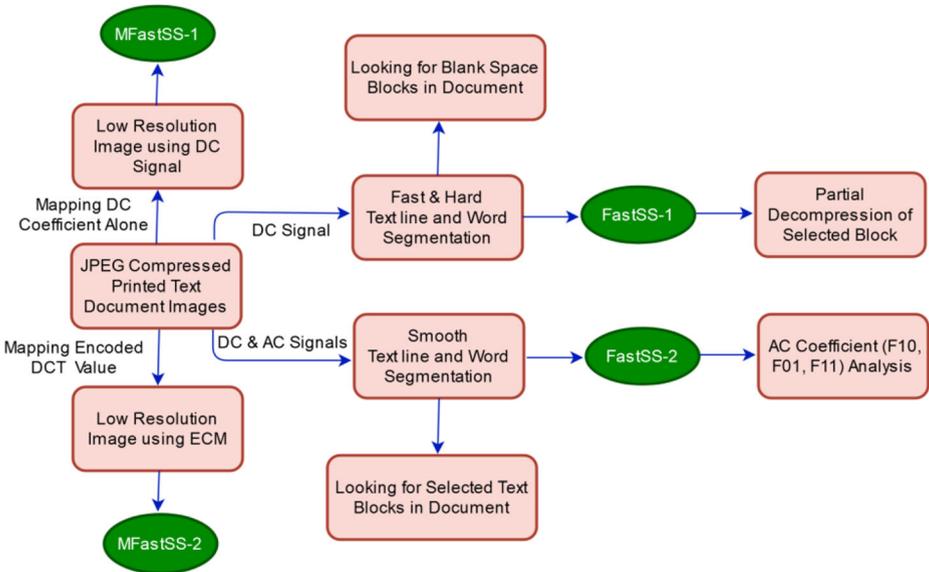


Fig. 1 Showing the summary of different contributions reported in this research paper. FastSS-1 and FastSS-2 are Fast and Smooth Segmentation algorithms that work directly with DC and AC signals. MFASTSS-1 and MFASTSS-2 are More Faster Segmentation algorithms that work on compressed images by extracting its low resolution version

2 Brief illustration of JPEG compression on printed text

For easy comprehension of the techniques proposed in this paper, this section briefly describes the working of a JPEG compression algorithm, and illustrates how a typical text document gets compressed when a JPEG algorithm is applied. Original image $f(i, j)$ in the pixel domain is divided into 8×8 non-overlapping blocks. Each block is then transformed using a forward Discrete Cosine Transform (DCT) into a frequency domain on a block by block basis. The DCT of an image in the pixel domain is obtained through Eq. (1). Quantization is performed on each 8×8 DCT coefficient block to discard visually insignificant coefficients, and due to which compression is achieved with the loss of high frequency information. The resulting quantized DCT blocks become sparse matrices where just few coefficients (commonly F_{00} , F_{01} , F_{10} , F_{20} , F_{11} , F_{02} at 50% quality factor) would be left at the top-left corner. Then a lossless entropy coding (in most cases, Huffman coding) is performed on DC and AC signals separately to get further compression [26, 41]. During decompression, the same steps (in reverse order) are applied to reproduce the image from its compressed stream. First the entropy coded data stream is decoded to get the quantized coefficients. Then dequantization is applied on these coefficients, and finally Inverse Discrete Cosine Transformation (IDCT) is applied to obtain the reconstructed image of the compressed image.

$$F_{uv} = \frac{C_u C_v}{4} \sum_{i=0}^7 \sum_{j=0}^7 \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} f(i, j) \tag{1}$$

Where, $C_u, C_v = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for } u, v \\ 1, & \text{otherwise,} \end{cases}$

A sample text image is shown in Fig. 2 to illustrate how JPEG compression is applied for compressing the pixel image. This figure shows the encoding of 8×8 DCT blocks when the DCT transformation is performed on the printed text document images (Figs. 3 and 4). In the figure, blocks within the pink color represent the base region where they cover the maximum text portion of a text-line in the document. Similarly the blocks between green (top) and pink (down) colors represent the ascender region. In this region, some of the text portions of the text-line come up into the above blocks of the base regions. Similarly the blocks within pink (top) and green (down) are called descender regions. Also here, some of the text-line contents come down into the below blocks of the base region. It is observed that each 8×8 DCT block may cover part of the character, almost full character or very few portions of character and so on (as shown in some cases in Fig. 5). There are also some blocks where no text parts are covered, and they are called space blocks. Some of the space blocks are found between adjacent text-lines which are called text-line separators (observe the blocks above and below the green line). Similarly there are also some blocks between adjacent words (observe blocks between blue lines). These blocks are called word separator space blocks. Most of the text-line separators (8×8 DCT space blocks) are appearing either in the ascender and descender regions, and these blocks may not be located in one complete row, so that row can be easily detected as text-line separating row, and without losing and bringing any extra text contents. In this case, when the projection is calculated, it is sure that one of these regions becomes an expected text-line separator. Therefore, in order to mark the exact text-line boundaries, the DCT blocks both in ascender and descender regions need to be analyzed. Similarly, for words also, just one or half word separating space blocks are available between characters of two adjacent words as shown in Fig. 2. In order to segment the word boundaries these separating space blocks should also be analyzed. Through the illustration of Fig. 2, the challenges of locating space blocks between text-lines and words are very clear, and therefore it is necessary to investigate novel algorithms for smooth extraction of text-line and word boundaries directly in the compressed file stream.

To better understand the above illustration, a practical analysis is provided. Here a sample printed JPEG compressed text document image is shown in Fig. 3(a) (displayed in uncompressed domain), and the corresponding one 8×8 pixel block

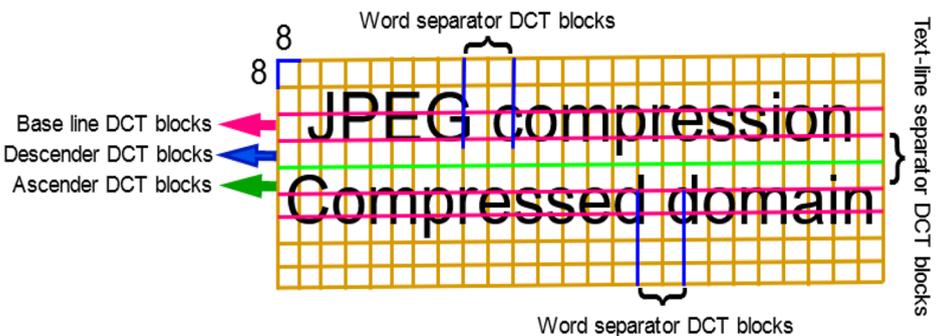


Fig. 2 A small text region of a sample document image shown in pixel domain illustrates how an 8×8 DCT block encodes printed text contents during JPEG compression, specifically at line separators and word separators, leaving no apparent clue for segmentation in the compressed data

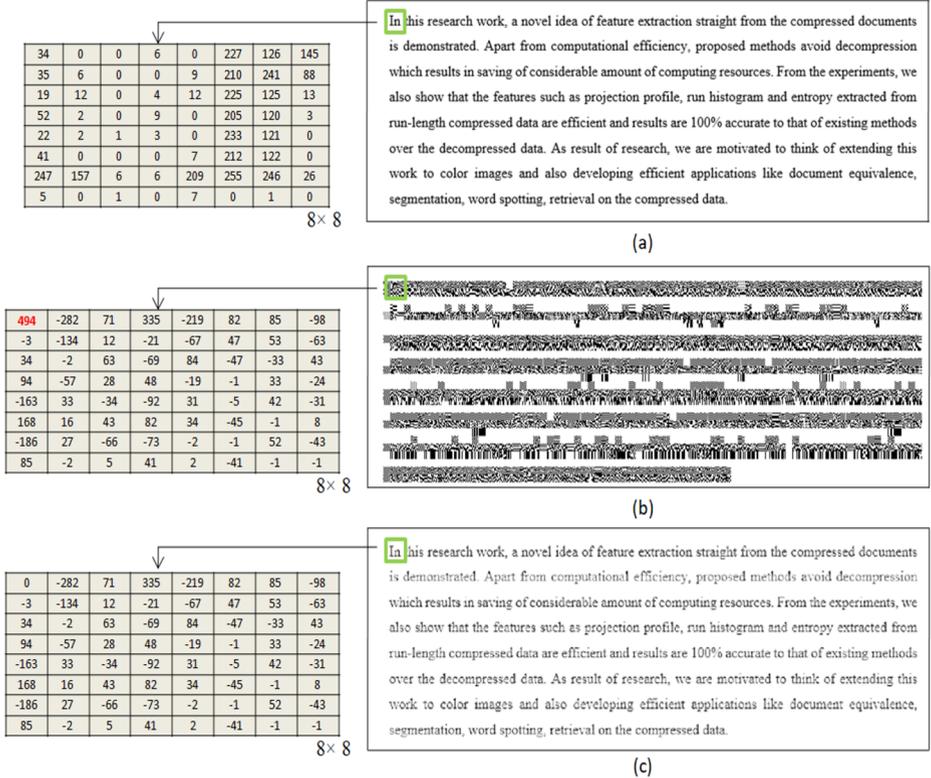


Fig. 3 A sample printed JPEG compressed document image (a) shown after decompression in pixel image from using both DC and AC coefficients. (b) Visualized in original DCT transformation from showing touching of text-lines, and further it provides no clue for the presence of words and word separator spaces. (c) Visualized after decompressing with only AC coefficients and leaving the DC signal

(highlighted with green color) of that image is shown on the left side. Similarly the DCT transformed stream of the same image and the corresponding one 8×8 DCT block is shown in Fig. 3(b). The reason for showing this figure is to depict how the text content will be arranged in the DCT compressed stream after applying the DCT transformation. In Fig. 3(b), it can be observed that adjacent text-lines are connected without leaving any separation gaps between them (observe line 4 and 5) for

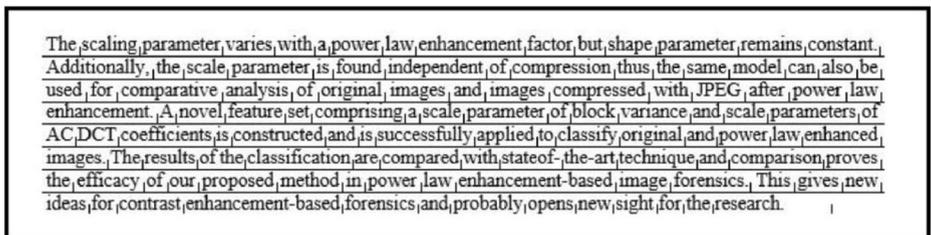


Fig. 4 A sample output image depicting how text-line and word boundaries can be easily segmented in the pixel domain

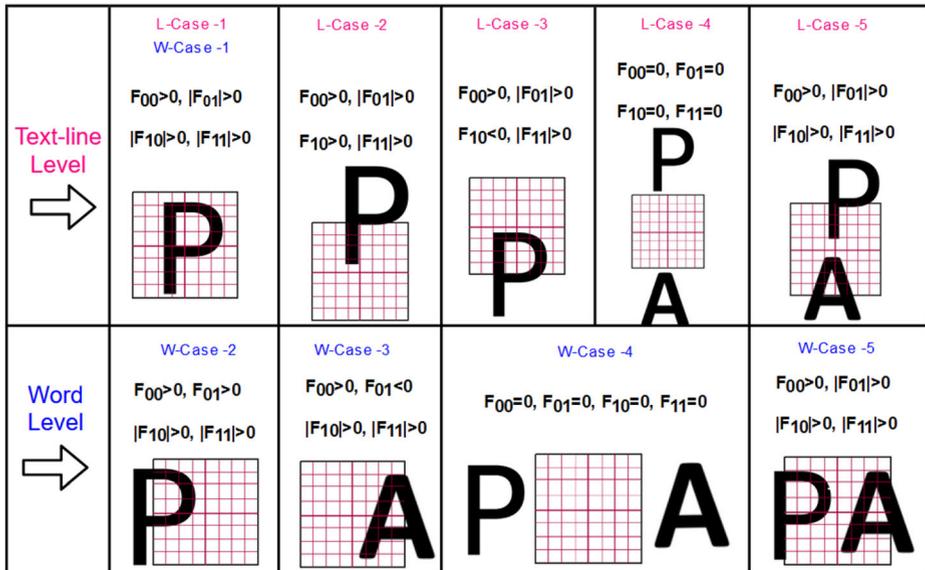


Fig. 5 The pictorial representation shows different block cuttings/encoding of the 8×8 DCT blocks during compression. At the top the cases for text-lines (pink color) and in the bottom for words (blue color) are shown. Behaviors of the corresponding AC coefficients for each DCT encoding are shown (black color)

segmentation. Similarly in case of words also, there is no visible clue for identifying the position of the words and also the separating gaps between any two adjacent words. However, the above image is created with 1.5 line spacing between text-lines. Imagine, if the image is in single line spacing, then the text contents would be closer, and smooth segmentation of text-lines and words in such documents become even more challenging than segmenting them from the earlier document.

Unlike the pixel image, there is no correlation clearly visualized among the coefficient values in the DCT block so that some information can be obtained. However, based on the procedure of DCT transformation explained above, some useful information about the pixel positions should be inferred to perform some applications in the compressed domain and to reflect them on the image when it is decompressed. Some useful insights are explained here. In the DCT block in Fig. 3(b) (left side), the first value (F_{00}) (red color) in the DCT block is called DC value (signal), and it gives the maximum information about the pixel density or the amount of content that particular block contains. Usually this value contributes the shade (with no frequency) information at the time of reconstruction. The whole image can be reconstructed just by using this one DC value, but this won't be sufficient to see the details of the text contents more clearly. Similarly, the remaining 63 coefficient values are called AC coefficients (signals). Each of these 63 coefficient values give the information about different frequency levels of pixels in that image. Since text documents are of high frequency, these frequency coefficient values and their signs are very important to infer some useful information. The whole image can be reconstructed with these 63 coefficient values with a reasonable quality. The reconstructed image in the uncompressed domain with all 63 AC coefficients leaving the

DC signal is shown in Fig. 3(c) (displayed in uncompressed domain), and the corresponding one 8×8 AC block coefficients are shown on the left side.

However, one 8×8 DCT block in the DCT stream is covering 8 rows of pixel height and 8 columns of pixel width. In case of printed text documents in the pixel domain, the text-line and word separating spaces between adjacent lines and words would be very small, and sometimes fall roughly within 3 or 4 rows for text-lines and 3 or 4 columns for words. Based on this information, and applying projection profile techniques, the smooth segmentation of text-line and word boundaries becomes an easy task in the uncompressed domain. A sample output image with text-line and word boundaries segmented smoothly in the uncompressed domain is shown in Fig. 4.

But finding such 3 or 4 lines and word separating space rows and columns (in case of pixels) in the DCT compressed stream is challenging. In a DCT compressed stream it needs at least 8 rows of space (in pixels) to identify the text-line separating space block. Similarly for words, at least 8 columns of space are needed to have one 8×8 DCT space block between two adjacent words. As there won't be any clue regarding how text gets compressed after applying the DCT transformation, segmentation of text-line and word boundaries in this domain becomes a challenging task. After observing the various standard document images, it is noticed that there exist at least 1 space block between adjacent text-lines and half space for words at different positions in the ascender and descender regions as also shown in Fig. 2. This may not be true and applicable for all kinds of documents. Different standard document images have been analyzed to identify the behavior of 8×8 DCT encoding over the text in these documents. The space block activity has been tabulated in Table 1. This table may be helpful for developing various applications in future directly in the JPEG compressed domain. Overall in this paper, since compressed representation is the input, there has to be an approach which can intelligently deal with these challenging problems for segmenting the text-line and word boundaries smoothly. The next section will describe the problem of text-line and word segmentation in detail.

Table 1 Demonstration of an average 8×8 DCT activity on various standard compressed printed text documents (font size and font style) involving JPEG compression algorithm

DPI	Document type	Image size	Font style	Font size	No of DCT block	
					In space	On char
96	Research article	160×124	TNR	10 pt	1	1
	News	120×134	Helvetica	12 pt	1.5	2
	Magazines	147×153	Garamond	12 pt	1.5	2
	Other blogs	142×121	Tahoma	12 pt	1.5	2
200	Research article	160×124	TNR	10 pt	1.5	2
	News	120×134	Helvetica	12 pt	1.5	2
	Magazines	147×153	Garamond	12 pt	1.5	2
	Other blogs	142×121	Tahoma	12 pt	1.5	2
300	Research article	160×124	TNR	12 pt	1.5	2
	News	120×134	Helvetica	12 pt	2	2
	Magazines	147×153	Garamond	12 pt	2	2
	Other blogs	142×121	Tahoma	12 pt	2	2

3 Problem description

Unlike text-line characters in the handwritten document, characters in the printed documents are very straight, uniform in leaving the space, containing no overlapping and skew. Therefore, as explained in Section 2, because of encoding the 8×8 region, it is certain that some characters would be encoded with separation spaces, and some parts of characters in adjacent text-lines and words come into one block. However, characters in English alphabets like a, c, e, i, m, n, and so on, are smaller in size, and comparatively characters j, g, p, b, d, f, h and so on, are larger in size. Moreover, in English printed alphabets, there are certain cases like different font size and font style, different types of spacing in case of a paragraph, text-line, word and characters.

Usually official documents have text-lines and words with single keyword spacing, and it is noticed by us that a single keyword space spans just one and half DCT blocks (at 300 DPI). During compression, 8×8 DCT block encoding for big letters and small letters would be different. Observe the text-lines in the base region of Fig. 2 where, 8×8 DCT encoding in case of big letter ‘p’ is observed in the base region and part of its portions is in the below region. Similarly, for another big letter ‘d’ 8×8 DCT encoding is seen in the base region and part of its portion is in the above region.

However, for small letters like ‘o’, the 8×8 DCT block encoding is almost within the 8×8 block size. Apart from these examples, the encoding of characters can be in any form. Therefore, segmentation of text-lines with the help of DCT blocks in the base region alone cannot be done and it loses some text-line contents or may bring extra contents from other text-lines (observe base region blocks in Fig. 2). For a smooth extraction, all the blocks above and below the base region are required to be analyzed, and the relationship of those blocks with the current and adjacent text-lines should be decided. During compression, there can be many combinations of encoding text contents and space regions between them. Some important combinations can be seen through Fig. 5 (Pink color represents the text-line cases and blue color represents the word cases). At some positions, due to less space between text-lines some part of the characters may be encoded (L-Case-2, L-Case-3), and at another positions because of sufficient separation space, a single 8×8 block can cut it and be as a text-line separating space block (L-Case-4); and sometimes only one block may cut the text from the both the text-lines which indicates no space between them (L-Case-5). Similarly, for word segmentation also, different cases can be observed in Fig. 5 (blue color), where a full 8×8 block may cover one letter (W-Case-1) or half of the letter (W-Case-2 and W-Case-3), and sometime letters from both the adjacent words (W-Case-5). In Order to segment text-lines and words, all the 8×8 DCT blocks need not be considered. With the detailed observation, the 8×8 DCT block is categorized into different cases which need to be analyzed are shown in Fig. 5. For each combination, the behavior of the AC signal F_{00} , F_{01} , F_{11} , F_{10} is shown for text-line and words separately. Except L-Case-4 and W-Case-4, which are empty space blocks, all 8×8 DCT blocks are required to be analyzed to recognize what type of block encoding it is, and thereby text-line and word boundaries can be decided as shown in the below cases.

$$Line = \begin{cases} above, & \text{if } F_{10} > 0 \\ below, & \text{if } F_{10} < 0 \\ both, & \text{if } |F_{11}| > 0 \end{cases} \quad word = \begin{cases} left, & \text{if } F_{01} > 0 \\ right, & \text{if } F_{01} < 0 \\ both, & \text{if } |F_{11}| > 0 \end{cases}$$

The challenges of text-line and word boundary detection with the analysis of DCT blocks are resolved through two strategies which are discussed in the following section.

4 Proposed model

Various stages in the proposed model are shown in Fig. 6. At the beginning DCT coefficients are extracted by applying partial decompression on the JPEG compressed printed text documents. As explained in Section 2, since the DC signal represents the average background shade of the block, during compression if an 8×8 block passes through text, then it will be reflected in the single DC coefficient value (F_{00}). Based on this observation, all the DC coefficients from each 8×8 DCT block are added horizontally to obtain the vertical DC projection profile as represented mathematically in Eq. (2).

$$DC_k = \sum_{i=1,8}^m \sum_{j=1,8}^n F(i,j) \quad (2)$$

Where, we used the standard block size of 8×8 .

The profile provides significant information about the position of text and the separation space. If $DC_k > 0$ indicates the existence of text in the block and $DC_k = 0$ indicates the presence of no text, which in other words hints for the separator position. However, as explained in Section 3, in many cases, the line separators for all the text-lines may not be clearly visible due to spacing, overlapping and font constraints. In such a case, the expected line separators are predicted using peaks of the DC projection profile that is obtained at the base region of each text-line. Mid-

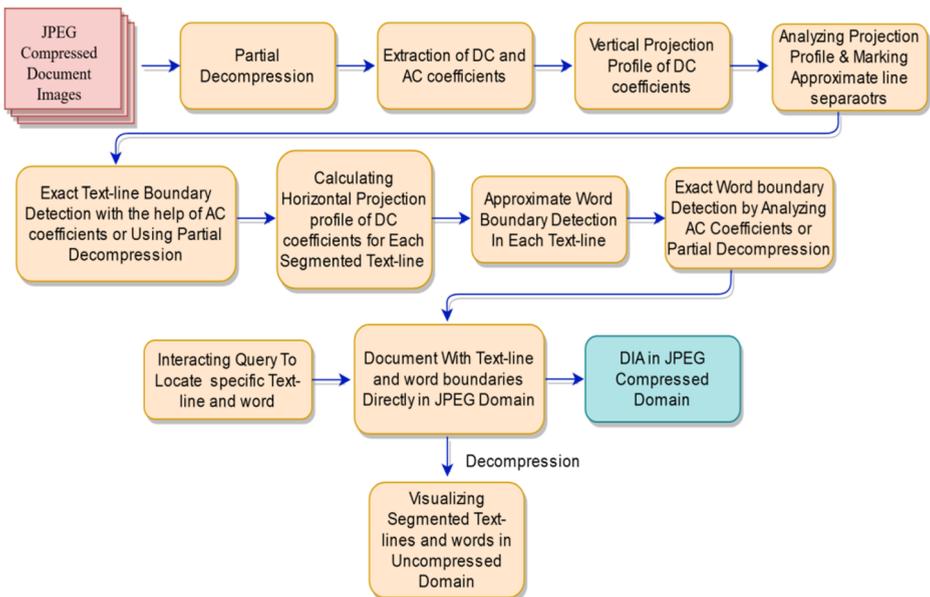


Fig. 6 Flow diagram of the proposed techniques for fast and smooth analysis of printed text-line and word segments directly in JPEG compressed domain

block between any two peaks is taken as expected text-line separator which is also called as hard segmentation, and the strategies (FastSS-1 and FastSS-2) to be discussed next are applied to detect the exact smooth boundary of the text-line.

Both F_{10} , F_{11} coefficients are needed for smooth analysis of text-line boundaries. Similarly for word segmentation also, after getting the exact line boundaries, the horizontal projection profile is calculated for each line. With the help of projection valleys, the approximate word boundaries are marked. Now, the exact word boundary is identified by processing the F_{01} , F_{11} coefficients from each block in the column of the expected word boundary. The two segmentation strategies, one is using partial decompression of selected blocks (FastSS-1), and second one is based on using AC coefficients (FastSS-2) and without involving any decompression are explained in detail below (Tables 2 and 3).

4.1 FastSS-1: partial decompression of selected blocks

After marking the expected text-line and word separators using the projection profile of DC signal, this strategy proposes to decompress only those 8×8 DCT blocks that are in the expected text-line and word separators, and make the decision regarding the boundary of the text-lines in the pixel domain by analyzing the vertical projection profile of pixels. Since this technique does decompression of only those 8×8 DCT blocks that are located along the expected line and word separators, and hence it is efficient in comparison to decompression of the entire document in pixel domain for processing (see Table 4). But in the case of L-Case-5 and W-Case-5 as shown in Fig. 4, this strategy, though it locates the exact horizontal partition between two characters, the boundary mapping will be in 8×8 blocks only. Therefore, in order to avoid any loss of information from both adjacent text-lines and words, such blocks will get boundary membership of both the text-lines and words.

4.2 FastSS-2: using AC coefficients and without decompressing the selected blocks

As already discussed, the AC coefficient values and their signs in the 8×8 DCT block are generated by the linear combination of all the 64 pixels in that block [34]. For example, the F_{10} coefficient shown mathematically in Eq. (3) and pictorially visualized in Fig. 7, is the summation of differences between pixel values in the first row from the eight row, second row from the seventh row and so on. Similarly, F_{01} coefficient value is the summation of differences between pixel values in the first four columns and the next four columns. Therefore, by using the F_{10} and F_{01} coefficients

Table 2 Demonstrating the significance and advantages of working with low resolution images, variation of the compressed document image with reduction in file size and processing time (hard segmentation)

Image size	DC_Reduced image size	ECM image size	DCT image size	Processing time on DC_Reduced image (sec)	Processing time on ECM image (sec)	Processing time on DCT image (sec)
600×624	53×70	53×70	424×560	0.00040	0.00043	0.1027
256×320	32×40	32×40	256×320	0.00021	0.00022	0.0339
256×664	32×83	32×83	256×664	0.00027	0.00033	0.0774

Table 3 The overall experimental results of all the proposed methods (FastSS-1, FastSS-2, MFastSS-1, MFastSS-2) for the extraction of text-line and word boundaries in JPEG compressed printed documents tested on self-created dataset. The results of the same operations in the pixel domain are also shown for comparison

Different methods	Image (DPI)	Text-line segmentation		Word segmentation		F-measure (%)	
		Precision	Recall (%)	Precision	Recall (%)	Text-line	Word
FastSS-1	96	92	100	80	100	95.83	88.88
	200	98	100	96	100	98.98	97.95
	300	100	100	100	100	100	100
FastSS-2	96	90	100	80	100	94.73	84.75
	200	95	100	91	100	97.43	95.28
	300	99	100	98	100	99.49	98.98
MFASTSS-1	96	90	100	80	100	94.73	88.88
	200	93	100	89	100	96.37	94.17
	300	99	100	98	100	99.49	98.98
MFASTSS-2	96	90	100	80	100	94.73	84.75
	200	93	100	89	100	96.37	94.17
	300	99	100	98	100	99.49	98.98
Pixel domain	96	100	100	99	100	100	99.5
	200	100	100	99	100	100	99.5
	300	100	100	100	100	100	100

the L-Case-2 and W-Case-1 as shown in Fig. 5 can be identified and based on their signs text-line and word boundaries can be resolved. Which means, unlike the handwritten text, in printed text-lines the text characters would be horizontally and vertically aligned. At the expected separators, it can be seen that the big sized letters occupy either the above portion or below portion of the 8×8 blocks. During compression, if text falls on the above portion of the block then the F_{10} coefficient in that block becomes positive, and if text falls on the below portion of block the F_{10} is becoming negative. Similarly, if text falls on the left side of the block then F_{01} becomes positive, and text falls on the right side of the block then F_{01} becomes negative.

Table 4 Comparison of the performance of the proposed methods (tested on 300 DPI documents) in the JPEG domain with other methods existing in various compressed domains

Methods	Parameters count		Text-line segmentation (%)		Word segmentation (%)		F-measure (%)	
	(Lines)	(Words)	(Precision)	(Recall)	(Precision)	(Recall)	(Precision)	(Recall)
Amamath [2] (runlength & handwritten)	2649	–	95.88	89.2	-	-	92.41	–
Rajesh [29] (JPEG & handwritten)	4034	29,717	98.40	96.7	96.3	99.41	97.54	97.83
Javed [16] (runlength & printed)	434	1409	99.09	100	96.96	99.54	99.43	98.18
Proposed (200 DPI) (JPEG & printed)	1073	8420	99	100	98	100	99.49	98.98

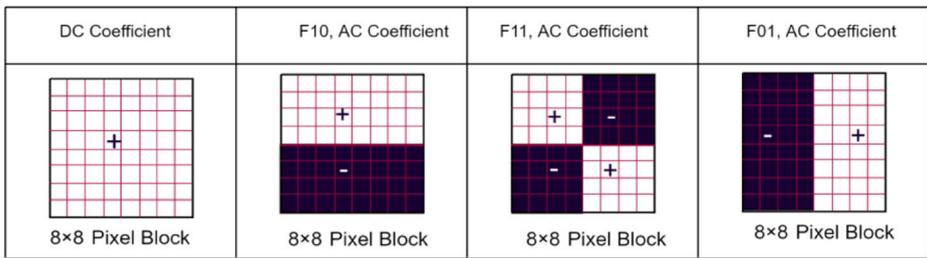


Fig. 7 Pictorial visualization of 8 × 8 DCT block AC coefficients F₀₀, F₁₀, F₁₁, F₀₁

$$F_{10} = \frac{c_1 c_0}{4} \left[\cos \frac{\pi}{16} \left(\sum_{j=0}^7 f(0,j) - \sum_{j=0}^7 f(7,j) \right) + \cos \frac{3\pi}{16} \left(\sum_{j=0}^7 f(1,j) - \sum_{j=0}^7 f(6,j) \right) + \cos \frac{\pi}{16} \left(\sum_{j=0}^7 f(2,j) - \sum_{j=0}^7 f(7,j) \right) + \cos \frac{7\pi}{16} \left(\sum_{j=0}^7 f(3,j) - \sum_{j=0}^7 f(4,j) \right) \right] \tag{3}$$

Therefore to resolve such a DCT block and to separate text-line and word boundaries it simply checks F₁₀ and F₀₁ coefficients. If F₁₀ > 0 then that character/block belongs to the above text-line, and if F₁₀ < 0 then that character/block belongs to the below text-line. Similarly, if F₀₁ > 0, then it belongs to the word on the left side, and if F₀₁ < 0, then it belongs to the word on the right side. But in the base region, this may not be the case every time, like in case of L-Case-5 and W-Case-5 as shown in Fig. 5. These cases will appear when the two characters of the adjacent text-lines or words are very close to each other, and the 8 × 8 pixel block will have partial characters from both the adjacent text-lines and words. Because characters are just like edges having high frequency, it is observed that letters in such blocks will be just like a sudden frequency change. In the basis function of a 8 × 8 DCT block, if tracing is done in a zig-zag fashion from left upper triangular matrix to right lower triangular matrix, the frequency level is in increasing order [26]. Therefore, in order to recognize blocks partially containing text contents of two adjacent text-lines and words, use of F₁₁ coefficient is proposed. As displayed in the L-Case-5 and W-Case-5 in Fig. 5, where above letter occupies the first four rows and below letter takes last four rows of pixels. Similarly for words also, the left side letter occupies the first four columns and the right letter occupies the last four columns. Here, there is sudden frequency change in both parts of the block. Using the F₁₁ coefficient the L-Case-5 and W-Case-5 blocks can be identified that are usually anticipated along the position of expected line and word separators. In order to identify the L-Case-5 and W-Case-5 cases in the approximate boundaries the absolute value of the F₁₁ coefficient is used to check the frequency in that block. If |F₁₁| > 0, then it confirms that the block has information from both the adjacent text-lines and words. However, because it is the summation of the difference between first four columns and next four columns of the first four rows (1-4 in pixel block) and difference between first four columns and next four columns of the next four rows (4-8 in pixel block), for L-Case-1, L-case-2, W-case-1 and W-case-2 also the |F₁₁| coefficient is getting some value. But in most cases |F₁₁| value is very high when L-Case-5 and W-Case-5 occur in the expected separators. In order to locate such blocks exactly, based on the experimental observation, the |F₁₁| value is set to be more than 50% of F₀₀ value. If a block along the expected line and word

separators contains such blocks, then we cannot have a hard partition for the exact text-line and word boundary between the adjacent text-lines and words. In order to avoid the loss of text information, the current block itself represents boundaries for both the text-lines and words. Therefore these selected blocks need to be decompressed to find the exact boundaries of text-lines and words using vertical and horizontal projection profiles of pixel values for text-line and words. This type of case occurs when there is not enough space block in the approximate separator. However an attempt has been made to avoid $|F_{11}| > 0$ decompression also, and perform the segmentation fully in the compressed domain. This process is explained in detail in below

4.2.1 Without decompression of F_{11}

This subsection explains the segmentation of text contents in case of L-Case-5 and W-Case-5 directly in the compressed domain. As explained above, In an 8×8 DCT block, if $|F_{11}| > 0$, then 8×8 DCT block contains text from both the adjacent text-lines as shown in Fig. 8(a) (which is the cross section view of L-Case-5), where text content is in black color and white color is the background of the document. On one side this block should be given to both the text-lines and at the same time content from the top text-line should not be there when it is moved to the bottom text-line and vice versa. This case, experimenting by changing various coefficient values and signs, proposed to use 70% of DC value to solve this segmentation problem. As explained above, the significance of the magnitude of the DC signal is that during compression it preserves the average background color which is suitable to that block, and F_{10} and F_{01} also preserve the first level of frequency which exists in the block. Therefore the DC value can be used to replace the F_{10} and F_{01} values to switch the colors of the top portion and bottom portion of the block for text-lines and left portion and right portion of the block for words. When F_{10} value is replaced by 70% positive DC value then the top portion is converted to background color of the document as shown in the Fig. 8(b). Similarly, if F_{10} value is replaced by 70% of negative DC value then the bottom portion of the block is converted to background color. However, since F_{10} and F_{01} do not give complete black and white shades at the middle (observe F_{10} and F_{01} positions in the basis image of JPEG compression in

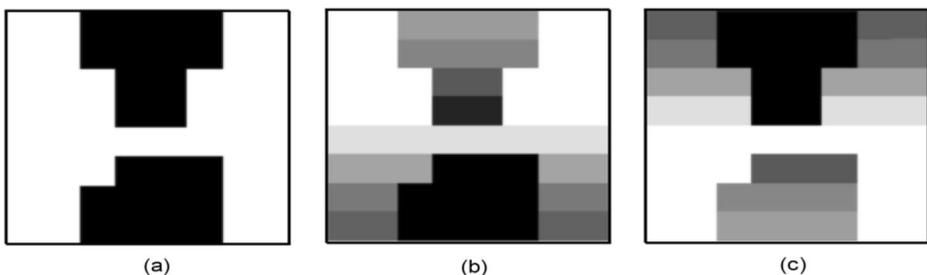


Fig. 8 Pictorial illustration of segmenting the text contents in L-Case-5 DCT block. (a) DCT block having text content from both lines, where black color represents the text content and white color represents the background of the document. (b) DCT block (top portion is converted as background) after replacing the F_{10} with 70% of F_{00} coefficient value. (c) DCT block (bottom portion is converted as background) after replacing the F_{10} with negative 70% of F_{00} coefficient value

[26]), certain shortcomings are observed in using F_{00} value in place of F_{10} F_{01} coefficients.

Algorithm 1 Smooth extraction of text-line boundaries in JPEG compressed stream using AC signal analysis. DCT_Img is mapped into the Well Separated Image (WSI) with clear separation gaps of text-lines and smooth segmentation of text-line boundaries.

Input: DCT_Image with approximate text-line separators.

Output: Smooth segmentation of text-line boundaries at approximate line separators using F_{10} and F_{11} coefficients.

1. Partial Decompression of JPEG compressed stream
 2. Placing of approximate text-line separators using vertical projection profile of DC signal
 3. **for** every 8×8 DCT block along the column do
 4. **for** every 8×8 DCT block along the row do
 5. Copy the 8×8 DCT block into WSI
 6. **if** text-line separator block is detected **then**
 7. **if** ($|F_{11}| > 0$) **then**
 8. Replacing the F_{10} value with 70% of F_{00} value
 9. Moving that block to above text-line in WSI
 10. Inserting text-line separating space block below
 11. Replacing F_{10} value with 70% of negative of F_{00} value
 12. Moving that block to below text-line in WSI
 13. Moving to next 8×8 block in next row in DCT
 14. **else**
 15. **if** ($F_{10} > 0$) **then**
 16. Copy that block into WSI
 17. Inserting text-line separating space block below
 18. Moving to next 8×8 block in next row in DCT
 19. **else if** ($F_{10} < 0$) **then**
 20. Move that block below text-line in the WSI
 21. Moving to next 8×8 block in next row in DCT
 22. **else**
 23. Moving to next 8×8 block in next row in DCT
 24. **end if**
 25. **end if**
 26. **else**
 27. Moving to next 8×8 block in next row in DCT
 28. **end if**
 29. **end for**
 30. Moving to next 8×8 block in next column in DCT
 31. **end** **for**
-

First shortcoming is, after decompressing that block the background color for the half portion is coming up with shades (more black to white shade) as can be observed at the top portion in Fig. 8(b) when F_{10} is replaced by negative of F_{00} in the case of text-line (Figs. 9 and 10). Because of that, after decompression, some text contents can be observed at the peripheral portions of the text-lines and words. The second side effect of using this in the FastSS-2 algorithm is that, during compression, the DCT encoding will not be limited to the half portion of the block alone. Encoding of text may cross more than half as can be observed in line separator 7 in Fig. 11(a). Because of this problem, although the text portion is from the same block, because of applying this strategy in the FastSS-2 algorithm the appearances of partial text contents at the peripheral portions of text-lines are increased. However, though there is no space between the adjacent text-lines and words, by using DC projection profile and F_{10} , F_{01} and F_{11} AC coefficients, and selected blocks decompression the compressed document can be analyzed to segment printed text-lines and words directly in the compressed domain with reasonable quality as shown in Fig. 11. The DC coefficients shall be further investigated to design the better segmentation approaches in the future. The text-line segmentation process based on using FastSS-2 technique is given in the form of an algorithm (see Algorithm 1). The algorithm processes documents from top to bottom and left to right in the compressed stream, and maps the text-contents into a new document called Well Separated Image (WSI) where all the text-lines are well separated with two full 8×8 space blocks for clear visualization of boundaries. When it comes to the approximate line separator block, it first checks F_{11} value, and if the absolute value is greater than 50% of DC value then it adjusts that block to both lines. Then it checks the sign of the F_{10} coefficient in that block. If it is negative, that means the character belongs to below text-line, and then it shifts that current block below by two blocks and replaces that current position with separation space. Similarly, if it is positive then it maps the current block there only and creates the space block below that block in the WSI. Finally if the block is empty, since it is a space block, then it does not do any operation. The output of this algorithm (WSI) is shown in Fig. 11(c) in the experiments section.

Some challenges have been identified when word segmentation algorithm has been applied on documents having smaller font sizes and not having enough separating space between words. In these documents, one 8×8 block is spanning one full

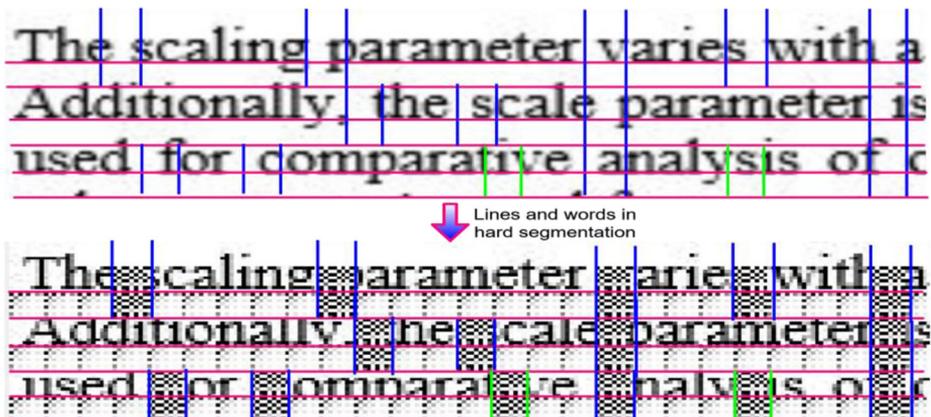


Fig. 9 Pictorial illustration to show the word boundaries (blue color) at the approximate word separators identified using horizontal projection profile. Due to low projection wrong word separators (green color) are calculated. Corresponding uncompressed image is shown at the top to visualize the corresponding characters



Fig. 10 Displaying the text-lines and words in low resolution images generated by further compressing the DCT compressed stream. **(a)** A sample image generated using DC_Reduced strategy. **(b)** A sample image generated using ECM strategy

character and sometimes two characters, if they are of smaller size in width and height. Due to this, the horizontal projection profile for word segmentation is not weighing appropriately so that word blocks and their separating space blocks can be seen clearly. It is because of two reasons. One is because of the DC value, which is calculated for the whole 8×8 pixel block, which get some value even when the 8×8 DCT block encodes the small portion of the text and large portion of the separating gaps between words. The second reason is when 8×8 DCT block encoding happens on the space within the characters. Because of this reason, some of the other areas in the middle of words are also marked as word separation gaps as shown in Fig. 9, where blue color represents the actual word gaps and green color represents the non-word separating gaps. Because of some of these practical challenges identifying the words and word separation gaps is becoming very difficult. However, rest of the word separation gaps which are marked correctly have been properly analyzed for the extraction of word boundaries smoothly. Some of the other segmentation approaches in the literature have been explored in the process of finding a better solution to these challenges, and though they could not solve these problems, they located expected separators in less time. they are explained in the below sections.

A JPEG compressed printed text document image decompressed with only DC values, and mapped to 0 (represents space) and 1 (represents text content) called as DC_Reduced Image as shown in Fig. 10(a). Another kind of representation called the Encoding Cost Map (ECM) is addressed in [8]. Here all the DCT blocks are decompressed, and each 8×8 DCT block is encoded to one single value and mapped on to the other file called as ECM Image. With this encoding the temporary image is generated as shown in the Fig. 10(b). Based on the activity of the block, most of the objects in the image are highlighted. Usually ECM representation is used for segmenting the larger objects in the document images. However, each representation of compressed stream makes a clear distinction with respect to practical analysis. With respect to DC_Reduced image, from each DCT block extracting DC value alone makes huge reduction in the image size. For example the DCT image of size 16×16 contains 4 DCT 8×8 block. Extracting one DC value from each block makes the DC_Reduced image of size 4×4 . There is 98.43% reduction in the size. Similarly apart from the encoding costs, same reduction would be there in the ECM image also. However, if 10 AC coefficients are kept then there would 85.29% reduction in the image size. Therefore analyzing the different representation of the image make huge difference in practical implementations. The detailed information about all these representations is shown in Table 2.

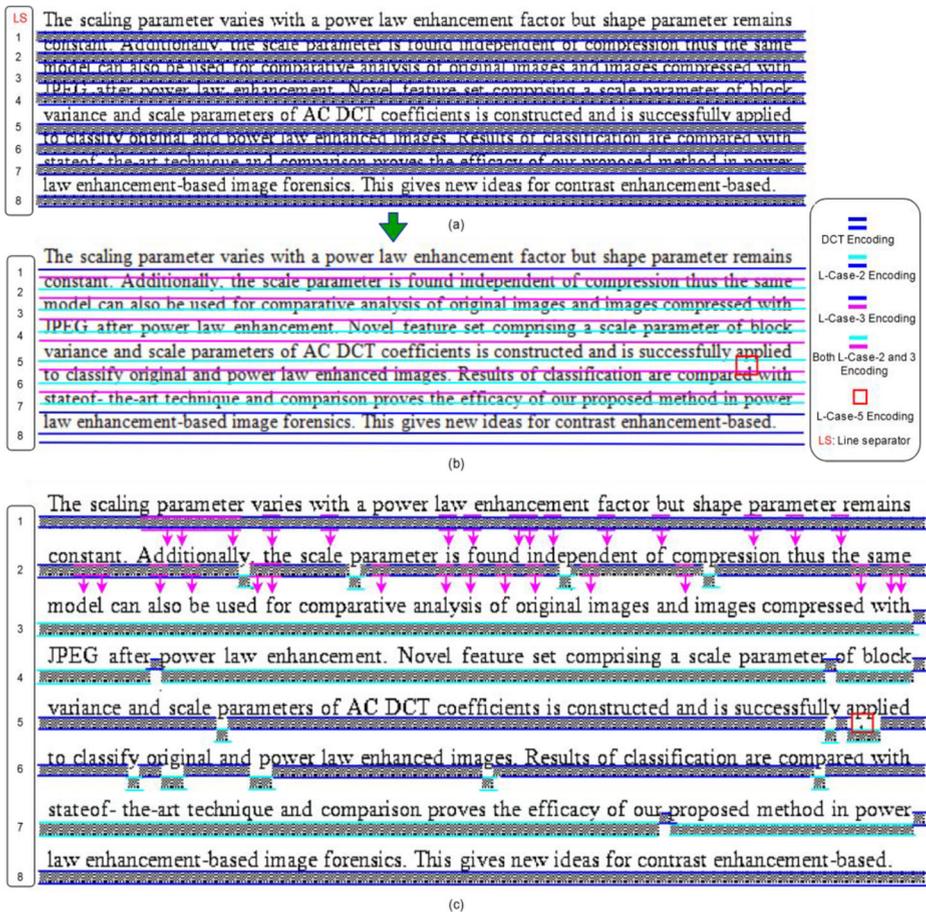


Fig. 11 Pictorial illustration of output image with smooth segmented and well separated text-lines after applying the FastSS-2 technique for analyzing the AC coefficients along the expected text-line separators. (a) Expected text-line separators (blue color) calculated using vertical projection of DC values in each 8×8 block in DCT image of size 168×640 (displayed in uncompressed domain). (b) Showing the various cases of 8×8 DCT encoding of text contents in the corresponding uncompressed image with the help of colors, text contents between magenta (top) and blue (bottom) contain the L-Case-2 encoding, blue (top) and pink (bottom) represents the L-Case-3 encoding, magenta (top) and pink (bottom) contains both L-Case-2 and L-Case-3 encodings, blocks between blue color does not have any encoding of text (line separator 8), and finally the portion in the red color shows the L-Case-5 case. (c) The final output image of size 272×640 with well separated text-lines, blue color shows the same expected text-line separators, pink color with arrows shows the previous and current positions L-Case-2 blocks replaced as space after sending the contents to below line (showed for first two text-lines only), similarly blocks within the magenta color shows the L-Case-3 blocks shifted down by one block and finally red color show the L-case-5 block

4.3 Approaches for more fast and smooth segmentation (MFastSS) of text-line and word boundaries

The advantages that we achieve due to compression of large volumes of data are better storage (low memory) and efficient transmission (less time). For instance image $f(i, j)$ is compressed with $x\%$ ratio using C compression, $CI = C(f)$ then compressed image $f'(k, l)$ size gets reduced to $x\%$ of its original size. Because of these reasons, according to compressed domain

processing, performing any computation on the compressed data takes comparatively less computation time and memory. In that process, if the compressed representation is further compressed then performing any operation on that data is even more efficient. For instance, If $f(k, l)$ has further reduced with $y\%$ then the total reduction of size is $(x + y + (x+y)/100)$. Two such approaches [8, 29] are considered from the literature to apply them for the segmentation of JPEG compressed documents into text-lines and words. Moreover, though these types of approaches in the literature are proposed for segmenting large objects, the proposed models in this paper are applied to achieve more faster and smoother segmentation of text-line and word boundaries.

4.3.1 DC_Reduced image technique (MFASTSS-1)

As explained in Section 3 that the DC value in a typical 8×8 block carries significant information about the content in the block. DC_Reduced is obtained by mapping each 8×8 DCT block into one value. Using this DC value the block activity can be estimated. Based on this fact, from a typical DCT image of size $f(i, j)$ all the DC values from each 8×8 DCT block are extracted and mapped onto the another file by preserving their positions and order as it is. This newly mapped file is called the DC_Reduced image [29]. Because of this mapping the original image of size $f(i, j)$ gets reduced to a low resolution of size $f(i/8, j/8)$. Upon this low resolution image the projection profile techniques are applied to detect the expected text-line and word separators. Because of locating the expected separators on the low resolution image, this method is more faster when compared to the DCT image. The DC_Reduced model in [29] is proposed for segmenting the text-lines and words in handwritten documents. But, since printed text-lines and words are very close in nature, and one full DCT block is replaced by just one DC value the DC_Reduced image of printed text document does not show the separation gaps clearly for estimating the clear boundaries of text-lines and words. A sample image of DC_Reduced image produced from Fig. 2 is shown in Fig. 10(a). After detecting the expected text-line and word separators, the low resolution image is mapped back to its DCT image for locating the exact boundaries based on using the FastSS-2 technique.

$$EC = \sum_{i=1}^8 \sum_{j=1}^8 |DCT(i, j)| \quad (4)$$

Where EC means Encoding Cost

4.3.2 ECM image technique (MFASTSS-2)

Similarly, instead of mapping the DC value directly, each 8×8 block from the DCT compressed representation is encoded to a single value [8]. This single value is mapped on to another file to represent each block. Which means that the absolute values of all the 64 coefficients are summed up to one value as shown in Eq. 4, where EC means Encoding Cost of DCT a block. This sum is again encoded into a total number of bits required to represent that number. For example the sum value 100 gets converted into 7 bits, using this system $100_{10} = 1100100_2$, and mapped onto the other file. The number of these total bits represents the block activity in the whole document. For example, due to JPEG's compression strategy, smooth areas produce the low ECM values, whereas edges generate the high ECM values. Wherever

high frequency signal is found in the document, then such blocks will turn out to be higher in bit length. This process is suitable and efficient for segmenting the large objects like text, non-text, graphical content and other regions on a typical document image. After ECM mapping the original image of size $f(i, j)$ is converted into a low resolution image of size $f(i/8, j/8)$. A sample ECM image of Fig. 4 is shown in Fig. 10(b). Except for the extra encoding cost for each 8×8 DCT block, the ECM mapping also converts the high resolution image into low resolution, and because of this locating the expected text-lines and word boundaries upon the low resolution images becomes faster compared to high resolution images. Once the expected boundaries are detected the low resolution image is mapped to the DCT image to get the exact boundaries of text-lines and words using the FastSS-2 technique.

5 Experiments and results

100 JPEG compressed document images are extracted by capturing various standard sources of print media like single column conference and journal papers, text books and magazines, similar to the work of [28]. The DPI (Dots Per Inch) and text spacing in the document images were manually adjusted to make it more complex and suitable for our requirement.

$$Precision = \frac{TP}{TP + FP} \times 100 \quad (5)$$

$$Recall = \frac{TP}{TP + FN} \times 100 \quad (6)$$

$$F = 2 \times \frac{Precision * Recall}{Precision + Recall} \times 100 \quad (7)$$

Total 300 images are created where 100 documents are there for each category (300 DPI, 200 DPI and 96 DPI). All the proposed methods have been tested on this dataset. However, three well known metrics which are Precision (P), Recall (R) and F-Measure (F), have been used to evaluate the performance of the proposed methods [30]. These metrics are also represented mathematically in Eqs. (5), (6) and (7) respectively. Where TP is the total number text-lines or words that are correctly segmented and FP is the number of text-lines and words that are false segmented. Similarly FN is the total number of text-lines or words that are not segmented. The experimental results of all the proposed models are shown in Table 3. The first approach (FastSS-1) which is through partial decompression of selected 8×8 DCT blocks along the expected text-line and word separators is shown in first row, and the experimental results using the second approach (FastSS-2) which involved the usage of AC coefficients such as F_{10} , F_{01} and F_{11} coefficients are shown in second row. Similarly, the third approach (MFastSS-1) is tested on the DC_Reduced images produced using the above dataset, and the results are shown in the third row. Finally, the results of the fourth approach, (MFastSS-2) tested on ECM images produced from the same dataset are reported in the fourth row. However, proposed models are compared with the other techniques reported in the literature. Proposed models are compared with the compressed domain techniques as shown in Table 4. Similarly, the proposed model is compared with the uncompressed domain techniques as shown in Table 5.

Table 5 Comparison of the performance of the proposed methods (tested on 300 DPI printed documents in JPEG domain) with existing methods in spatial domain

Methods	Parameters count		Text-line segmentation (%)		Word segmentation (%)		F-measure (%)	
	(Lines)	(Words)	(Precision)	(Recall)	(Precision)	(Recall)	(Precision)	(Recall)
Alaei [1]	4034	-	98.35	98.76	-	-	98.55	-
Fernández-Mota [9]	4034	-	98.40	95.00	-	-	96.67	-
Papavassiliou [27]	1771	13,311	98.46	98.20	93.57	92.46	98.33	93.01
Renton [30]	-	-	94.90	88.10	-	-	91.30	-
Louloudis [24]	1773	1331	97.40	97.40	93.90	90.80	97.40	92.30
Ryu [32]	-	-	98.64	98.68	-	-	98.66	-
Arivazhagan [3]	78,902	-	-	-	-	-	98.81	-
Proposed (JPEG) (compressed domain)	1073	8420	99	100	98	100	99.49	98.98

In all the methods, it was observed that when the dpi of the text document increased, the size of the text and the gaps between them increased, and due to which the DC projection profile produced at least one space block between text-lines and words. When there is enough space (at least one 8×8 space block) between contents, it locates the exact boundary with projection profiles itself. Then the analysis of AC coefficients is not needed. However, for a typical input image $f(i, j)$ containing, for example, P text-lines and Q words on average per line takes $O(i \times j)$ time to generate a Well Separated Image. But in the DCT domain, since the algorithm outputs a Well Separated Image by copying the whole 8×8 block at a time, the time complexity for copying all the blocks onto the Well Separated Image reduces to $O(i/8 \times j/8)$. In addition to that, it only takes a constant time to check coefficient signs in the blocks at each expected text-line and word separators to decide the exact text-line and word boundaries. The total constant time for both text-line and words are calculated approximately as $((P \times (j/8) + P \times Q \times (j/8))$. A sample image is created in Microsoft Word with a single line spacing option between text-lines, font size 12, Times New Roman font style and 96 DPI. The text-line segmentation output of that image is shown in Fig. 11(c), which is called a well separated image where all the text-line and their boundaries are well separated. To avoid any sort of confusion and to understand various cases of 8×8 DCT encodings, the overall output has been illustrated with the help of two more figures, Fig. 11(a) and (b). In the figure, the expected line separator 8×8 DCT blocks on the input image Fig. 11(a) (displayed in the uncompressed domain) are shown within the blue color lines. This is also called hard segmentation, where DCT blocks encode both text contents and separation spaces exist between contents. Here, different kinds of DCT encodings (as already shown in Fig. 5) can be observed in Fig. 11(b).

The blocks within the blue (top) and pink (bottom) colors represents the line separator having L-Case-3 DCT blocks (observe line separator 1 in Fig. 11(b)), and the blocks within the magenta (top) and blue (bottom) colors represents line separator having the L-Case-2 DCT blocks. Similarly, the blocks within the magenta (top) and pink (bottom) colors represent line separators having DCT blocks of both L-Case-3 and L-Case-2 blocks. Finally a block within the red square in line separator 5 represents the L-Case-5 DCT block. Well separated text-lines and exact line separators based on using the second approach (FastSS-2) are shown in Fig. 11(c) (displayed in uncompressed domain), where L-Case-3 blocks are moved to bottom line (indicated by pink color arrows for just first two text-lines) and replaced that block with text-

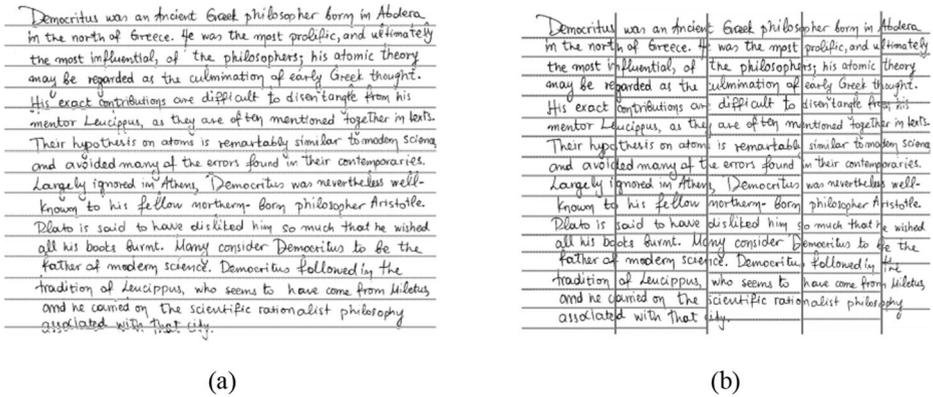


Fig. 12 Sample outputs of the proposed method for JPEG compressed printed text documents tested on JPEG compressed handwritten documents. (a) Text-line boundaries in a handwritten document (028.tiff, in ICDAR2013 dataset) after applying the FastSS-2 technique. (b) Text-line boundaries after applying the FastSS-2 method on the same image decomposed into vertical stripes to overcome the problem of skew

line separator space blocks. Similarly, L-Case-2 blocks remain in the same place but the below blocks are marked as text-line separator space blocks. Except L-Case-5, all blocks are smoothly segmented based on the analysis of AC coefficients. However, we admit that the coefficient based analysis may fail with handwritten and noisy images, in order to test the robustness of the proposed algorithms, the model is experimented with a sample handwritten document, and a sample output is shown in Fig. 12(a).

Due the presence of skew and touching, expected text-line separators between two adjacent text-line using projection profile techniques are not inserted properly, and some of them have fallen on the text contents. The skew problem can be reduced by using the piecewise projection profile technique to insert the text-line separators between the adjacent text-lines as shown in Fig. 12(b) [29]. Similarly, the proposed algorithms are tested on the printed documents containing noise. Here, 5% of Salt & and Pepper noise is added to the input image as shown in Fig. 13(a). It can be observed that the proposed algorithms are able to segment the text-lines in the noisy documents also, as shown in Fig. 13(b). Because of the presence of noise in some of the 8×8 pixel blocks, during compression of such noisy documents the coefficient values and signs may be changed. Because of that and in comparison to Fig. 11(c), more blocks in Fig. 13(b) have changed their positions (observe last line separator in Fig. 13(b)). Similarly, the noisy handwritten image is also tested on the proposed model. The noisy input image and segmented text-lines of the

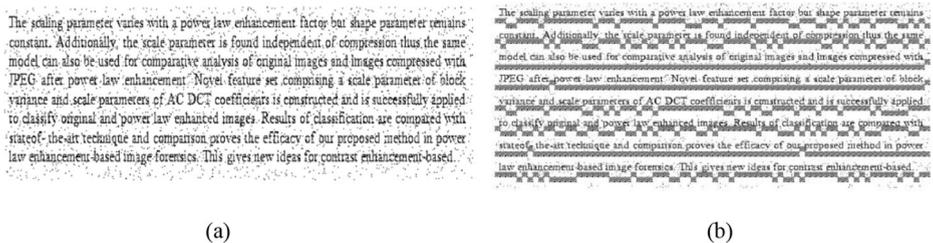
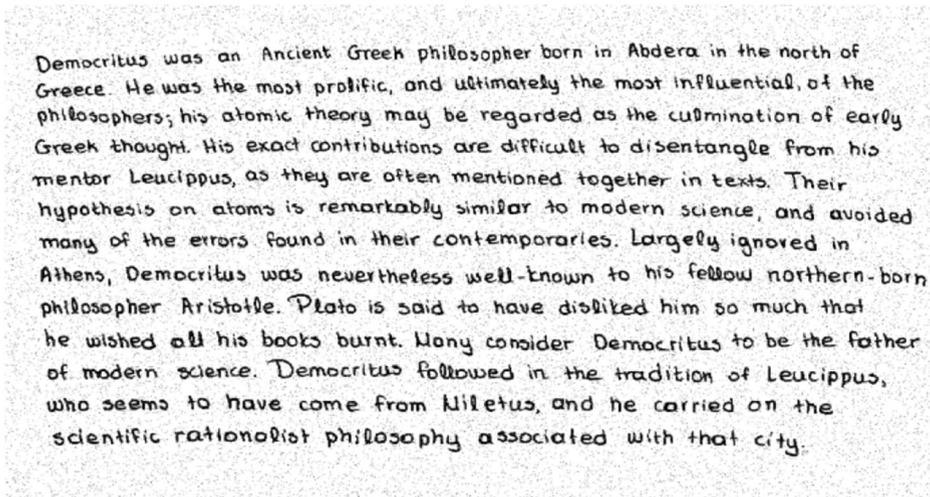
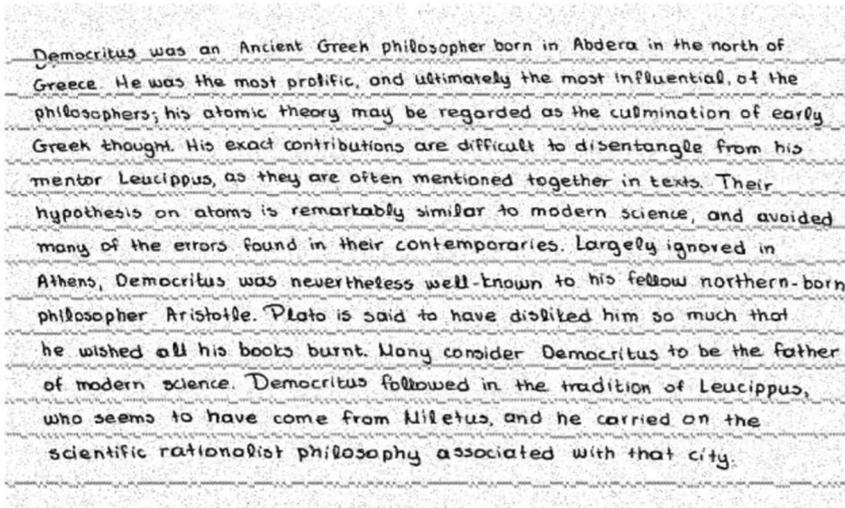


Fig. 13 Sample outputs of the proposed method tested on the printed text and noisy image. (a) JPEG compressed printed text document image with 5% of Salt & Pepper noise. (b) Text-line boundaries after applying the FastSS-2 method on the noisy JPEG compressed printed text document

noisy image are shown in Fig. 14(a) and (b). However, since it focuses more on the edges of the characters in the expected separators, the proposed model performed well on the documents having different font styles and large fonts. Moreover, Because of using projection profile techniques, the proposed algorithm does not work on more tilted documents. Similarly, proposed algorithms do not work on the documents with very low font sizes, where two characters may come into the same block. In this case, segmentation based on using the proposed algorithms cannot be done.



(a)



(b)

Fig. 14 Sample outputs of the proposed methods tested on the handwritten text and noisy image. (a) JPEG compressed handwritten text document image with 5% of Salt & Pepper noise. (b) Text-line boundaries after applying the FastSS-2 method on the noisy JPEG compressed handwritten text document image3)

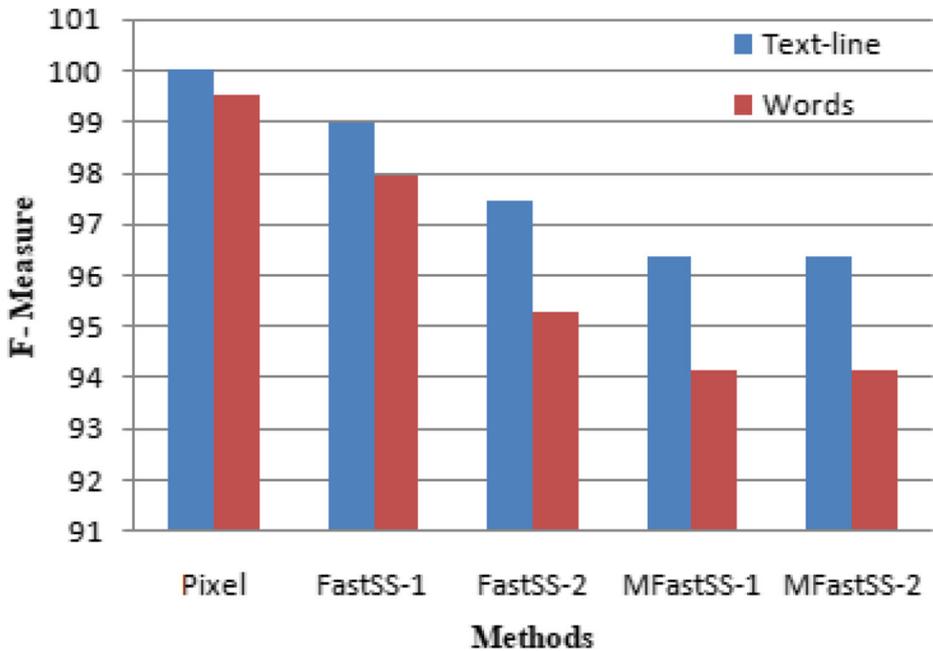


Fig. 15 A graph plot of the performance of the proposed methods for the extraction of text-line and word boundaries in JPEG compressed printed document image along with pixel domain result

However this is not the case with standard documents, where at least half or full space blocks are available. Performing segmentation just by checking a few AC coefficient values in a 8×8 DCT block, the proposed algorithms work faster in comparison with conventional decompression based text-line segmentation methods. For a typical document of size 600×624 pixels, 96 dpi size, and 27 text-lines and 414 words, we tested our algorithm in MatLab 2019a software with system configurations of 8GB RAM, i5 processors. The segmentation performance of the proposed methods are shown graphically in Fig. 15. Similarly, the efficacy in terms of execution time of each proposed model, and corresponding speed gained through the proposed methods with respect to pixel domain are shown in Table 6. It is found that the proposed algorithm performs much faster than the existing conventional methods and has achieved the segmentation accuracy more than 90% directly in the JPEG compressed domain.

Table 6 Execution times and speed gains of the proposed methods for the extraction of text-line and word boundaries in the JPEG compressed domain with respect to pixel domain

Different methods	Text-line & word segmentation time (sec)	Speed gain (%)
Pixel domain: decompression + processing	0.2027	–
FastSS-1: using partial decompression (DCT image)	0.0144	92.89
FastSS-2: using AC coefficients (DCT image)	0.00110	99.45
MFastSS-1: using AC coefficients (DC_reduced image)	0.00074	99.63
MFastSS-2: using AC coefficients (ECM image)	0.00099	99.51

6 Conclusion

This paper has investigated the possibility of Fast and Smooth Segmentation of text-line and word boundaries in JPEG compressed printed text document images. The compressed stream is extracted using partial decompression. The approximate text-line and word separators are marked by taking the projection profile of DC signal in each 8×8 DCT block. These approximate separators are further analyzed to segment the text-line and word boundaries smoothly based on using two proposed methods (FastSS-1 and FastSS-2). First method (FastSS-1) is based on partial decompression of selected DCT blocks. Second method is based on using direct AC coefficient values and without involving any decompression. Similarly two further faster methods (MFastSS) are devised on the low resolution images obtained by mapping the compressed stream. The first technique is developed on DC_Reduced image, and second on is developed on ECM image. Because of performing segmentation directly in the compressed domain, based on the experimental results, the proposed techniques have accomplished the segmentation with low computation time.

However, we are sure that the proposed JPEG compressed domain methods shall be utilized for realizing significant applications like word spotting, OCR and retrieval at faster rate. Finally, this would also open scope for developing many transform coding techniques and limitations of domain, and this would be our future work.

References

1. Alaei A, Pal U, Nagabhushan P (2011) A new scheme for unconstrained handwritten text-line segmentation. *Pattern Recogn* 44(4):917–928
2. Amarnath R, Nagabhushan P (2018) Text line segmentation in compressed representation of handwritten document using tunneling algorithm. *Int J Intell Syst Appl Eng* 6(4):251–261
3. Arivazhagan M, Srinivasan H, Srihari S (2007) A statistical approach to line segmentation in handwritten documents. *Proc. SPIE 6500, Document Recognition and Retrieval XIV 65000T*. <https://doi.org/10.1117/12.704538>
4. Bhowmik S, Sarkar R, Nasipuri M, Doermann D (2018) Text and non-text separation in offline document images: a survey. *IJDAR* 21(1–2):1–20
5. Boulid Y, Souhar A, Elkettani MY (2015) Arabic handwritten text line extraction using connected component analysis from a multi agent perspective. *IEEE Proceedings of International Conference on Intelligent Systems Design and Applications*, pp 80–87
6. Chebil F, Hadj Miled MKB, Islam A, Willner K (2005) Compressed domain editing of jpeg2000 images. *IEEE Trans Consum Electron* 51(2):710–717
7. Das D, Nayak D, Dash R, Majhi B, Zhang Y-D (2020) H-wordnet: a holistic convolutional neural network approach for handwritten word recognition. *IET Image Process* 14(9):1794–1805
8. de Queiroz RL, Eschbach R (1998) Fast segmentation of the jpeg-compressed documents. *J Electron Imaging* 7(2):367–378
9. Fernández-Mota D, Lladós J, Fornés A (2014) A graph-based approach for segmenting touching lines in historical handwritten documents. *Int J Doc Anal Recognit* 17(3):293–312
10. Florea C, Gordan M, Orza B, Vlaicu A (2013) Compressed domain computationally efficient processing scheme for jpeg image filtering. *Adv Eng Forum* 8-9:480–489
11. Frinken V, Fischer A, Manmatha R, Bunke H (2012) A novel word spotting method based on recurrent neural networks. *IEEE Trans Pattern Anal Mach Intell* 34(2):211–224
12. Giotis AP, Sfikas G, Gatos B, Nikou C (2017) A survey of document image word spotting techniques. *Pattern Recogn* 68:310–332. <https://doi.org/10.1016/j.patcog.2017.02.023>
13. Gueguen L, Sergeev A, Kadlec B, Liu R, Yosinski J (2018) Faster neural networks straight from jpeg. *NIPS18: Proceedings of the Advances in Neural Information Processing Systems*, pp 3933–3944

14. Hopkins M, Mitzenmacher M, Wagner-Carena S (2018) Simulated annealing for jpeg quantization. *IEEE Proceedings of Data Compression Conference*, pp 412–412
15. Hsin H, Sung T (2011) Image segmentation in the jpeg2000 domain. *IEEE Proceedings of International Conference on Wavelet Analysis and Pattern Recognition*, pp 24–28
16. Javed M, Nagabhushan P, Chaudhuri BB (2013) Extraction of line-word-character segments directly from run-length compressed printed text-documents. *IEEE Proceedings of National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*, pp 1–4
17. Javed M, Nagabhushan P, Chaudhuri BB (2015) A direct approach for word and character segmentation in run-length compressed documents with an application to word spotting. *IEEE Proceedings of International Conference on Document Analysis and Recognition*, pp 216–220
18. Javed M, Nagabhushan P, Chaudhuri BB (2018) A review on document image analysis techniques directly in the compressed domain. *Artif Intell Rev* 50(4):539–568. <https://doi.org/10.1007/s10462-017-9551-9>
19. Javed M, Nagabhushan P, Chaudhuri BB, Singh SK (2019) Edge based enhancement of retinal images using an efficient jpeg-compressed domain technique. *J Intell Fuzzy Syst* 36(1):541–556
20. Kasturi R, O’gorman L, Govindaraju V (2002) Document image analysis: a primer. *Sadhana* 27(1):3–22
21. Kidwai NR, Khan E, Reisslein M (2016) Zm-speck: a fast and memory less image coder for multimedia sensor networks. *IEEE Sensors J* 16(8):2575–2587
22. Kiumarsi E, Alaei A (2018) A hybrid method for text line extraction in handwritten document images. *IEEE Proceedings of International Conference on Frontiers in Handwriting Recognition*, pp 241–246
23. Kumar S, Gupta R, Khanna N, Chaudhury S, Joshi SD (2007) Text extraction and document image segmentation using matched wavelets and mrf model. *IEEE Trans Image Process* 16(8):2117–2128
24. Louloudis G, Gatos B, Pratikakis I, Halatsis C (2009) Text line and word segmentation of handwritten documents. *Pattern Recogn* 42(12):3169–3183
25. Lu Y, Tan CL (2003) Document retrieval from compressed images. *Pattern Recogn* 36(4):987–996
26. Mukhopadhyay J (2011) *Image and video processing in the compressed domain*. Chapman and Hall, Boca Raton
27. Papavassiliou V, Stafylakis T, Katsouros V, Carayannis G (2010) Handwritten document image segmentation into text lines and words. *Pattern Recogn* 43(1):369–377
28. Rajesh B, Javed M, Nagabhushan P (2019) Automatic text line segmentation directly in jpeg compressed document images. *IEEE Proceedings of Global Conference on Consumer Electronics*, pp 1067–1068
29. Rajesh B, Javed M, Nagabhushan P (2020) Automatic tracing and extraction of text-line and word segments directly in jpeg compressed document images. *IET Image Process* 14(9):1909–1919
30. Renton G, Soullard Y, Chatelain C, Adam S, Kermorvant C, Paquet T (2018) Fully convolutional network with dilated convolutions for handwritten text line segmentation. *Int J Doc Anal Recognit* 21(3):177–186
31. Retraint F, Zitzmann C (2020) Quality factor estimation of jpeg images using a statistical model. *Digit Signal Process* 103(102759):102759. <https://doi.org/10.1016/j.dsp.2020.102759>
32. Ryu J, Koo HI, Cho NI (2014) Language-independent text-line extraction algorithm for handwritten documents. *IEEE Signal Process Lett* 21(9):1115–1119
33. Shen B, Sethi IK (1996) Convolution-based edge detection for image/video in block dct domain. *J Vis Commun Image Represent* 7(4):411–423
34. Shen B, Sethi IK (1996) Direct feature extraction from compressed images. In: *Storage and retrieval for still image and video databases IV*. vol. 2670, pp 404–415. <https://doi.org/10.1117/12.234779>
35. Smith BC, Rowe LA (1993) Algorithms for manipulating compressed images. *IEEE Comput Graph Appl* 13(5):34–42
36. Song Q, Xiong R, Fan X, Liu X, Huang T, Gao W (2018) Compressed image restoration via external-image assisted band adaptive pca model learning. *IEEE Proceedings of Data Compression Conference*, pp 97–106
37. Tausif M, Kidwai NR, Khan E, Reisslein M (2015) Frwf-based lmbtc: memory-efficient image coding for visual sensors. *IEEE Sensors J* 15(11):6218–6228
38. Tausif M, Jain A, Khan E, Hasan M (2020) Low memory architectures of fractional wavelet filters for low-cost visual sensors and wearable devices. *IEEE Sensors J* 20(13):6863–6871
39. Tompkins DA, Kossentini F (1999) A fast segmentation algorithm for bi-level image compression using jbig2. *IEEE Proceedings of International Conference on Image Processing*, pp 224–228
40. Vo QN, Lee G (2016) Dense prediction for text line segmentation in handwritten document images. *IEEE Proceedings of International Conference on Image Processing*, pp 3264–3268
41. Wallace GK (1992) The jpeg still picture compression standard. *IEEE Trans Consum Electron* 38(1):xviii–xxxiv
42. Yanikoglu BA, Sandon PA (1998) Segmentation of off-line cursive handwriting using linear programming. *Pattern Recogn* 31:1825–1833
43. Yousfi Y, Fridrich J (2020) An intriguing struggle of cnns in jpeg steganalysis and the oneHot solution. *IEEE Signal Process Lett* 27:830–834

44. Zhang Y, Zha J, Chao H (2018) Fast h. 264/avc to hevvc transcoding based on compressed domain information. IEEE Proceedings of Data Compression Conference, pp 207–216
45. Zhong Y, Zhang H, Jain AK (2000) Automatic caption localization in compressed video. IEEE Trans Pattern Anal Mach Intell 22(4):385–392
46. Zhu N, Shen J, Niu X (2019) Double jpeg compression detection based on noise-free dct coefficients mixture histogram model. Symmetry 11(9):1119. <https://doi.org/10.3390/sym11091119>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Bulla Rajesh¹ • Mohammed Javed¹ • P. Nagabhushan¹

Bulla Rajesh
rajesh091106@gmail.com

P. Nagabhushan
pnagabhushan@iitit.ac.in

¹ Department of Information Technology, Indian Institute of Information Technology Allahabad, Prayagraj, U.P., India