Compressing the background layer in compound images, using JPEG and data filling

George Pavlidis\textsuperscript{a}, Christodoulos Chamzas\textsuperscript{b,\textsuperscript{*}}

\textsuperscript{a}Cultural & Educational Technology Institute, 58 Tsamiski Str., 67100 Xanthi, Greece
\textsuperscript{b}Electrical and Computer Engineering, Democritus University of Thrace, Vas. Sofias 12, 67100 Xanthi, Greece

Abstract

To efficiently compress rasterized compound documents, an encoder must be content-adaptive. Content adaptivity may be achieved by employing a layered approach. In such an approach, a compound image is segmented into layers so that appropriate encoders can be used to compress these layers individually. A major factor in using standard encoders efficiently is to match the layers' characteristics to those of the encoders by using data filling techniques to fill-in the initially sparse layers. In this work we present a review of methods dealing with data filling and propose also a sub-optimal non-linear projections scheme that efficiently matches the baseline JPEG coder in compressing background layers, leading to smaller files with better image quality.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Image coding; Image compression; Mixed raster content; Layered image coding; JPEG

1. Introduction

It is long recognized that the compression of images is content dependent. Today, a wealth of compression schemes is available to effectively deal with different kinds of images according to the characteristics of their content: there is bi-level image compression, continuous-tone image compression, as well as artificial image (graphics) compression. The most difficult case is that of the compound images, and that is because compound images may comprise all kinds of the aforementioned images. Since efficient compression is specific to each kind of image, an effective way of dealing with such images is the layered approach. According to the layered approach, a compound image is segmented according to some features into three or more layers. These layers can then be efficiently compressed by standard
encoders, such as JPEG [22,35,44] or JPEG2000 [23,43,26] for color layers and JBIG [21] or JBIG2 [20] for binary layers. This way the overall image compression and quality are optimized in comparison with the one-scheme-overall-compression approach.

Many works [1 6.8 17,19,25,27,29 33,36,39, 42,45 47] have already pointed out how this approach leads to significantly better results both in compression and final image quality. Basically, all techniques involve the separation of an image into layers, adopting a three-layer representation model, consisting usually of a binary mask, that captures the high frequency content (such as text and graphics), a foreground layer, that contains the color information for the masked pixels, and a background layer with the rest of the pixels, representing low-frequency content and smooth continuous tone image data. The purpose of separating an image into layers is to be able to compress each layer individually with a different and mostly appropriate compression scheme so that overall compression performance and quality are optimized.

The three-layer representation of a compound RGB color image leads to an augmentation of the data bit depth from 24 bpp to 49 bpp, since the representation involves two 24 bpp images (foreground/background) and one 1 bpp image (mask). Although the scheme seems extremely redundant, it is in this representation that lays the key to efficient compression. If the segmentation is “good” enough then the three layers can be compressed individually with very high efficiency and be of high quality. Though the model seems simple, it has proven to be extremely powerful, and, in minor variation, is already successfully employed in several commercial products, such as DjVu [7] and Digipaper [18]. In these cases the model, although based on the same idea, is not compliant to any standard. It is since the release of ITU MRC Rec. T.44 (Mixed Raster Content) [24], that the representation of a decomposed compound document image got a standard form. Based on this Rec., Sharpe and Buckley proposed a layered imaging architecture for the JPEG2000 compression scheme [38] in 2000, Mukherjee et al. proposed a fully MRC-compliant JPEG-matched compression scheme in 2001 [32] and a fully MRC-compliant JPEG2000-matched scheme in 2002 [31]. In [27], Jung and Seiler proposed an integrated segmentation and JPEG2000 compression system for compound documents. In [14] de Queiroz et al. based on prior work by de Queiroz [12,13] provided a system for the efficient compression of foreground and background data. In either case, compression of compound images can be divided into the following tasks:

- segmentation and production of a binary mask image,
- data filling of sparse foreground image,
- data filling of sparse background image,
- compression of each layer with an appropriate encoder,
- package of the whole representation into one meaningful stream for transmission or storage.

It has soon been realized that, since one of the consequences of the segmentation is that the produced layers are actually sparse matrices, there has to be an algorithm to fill-in the missing parts, so that existing compression schemes could actually operate efficiently on them. There are many techniques available for this task, although some of them did not initially appear as solutions to this problem. An important aspect is that the approach of filling-in the missing parts should be “compatible” with the characteristics of the encoder that is to be employed for each layer. Thus, the foreground and background layers have to be filled with specific color information, usually deduced from existing parts of the corresponding images, so that some sort of “continuity” is ensured in all areas of the resulting images that will be compressed with standard lossy/lossless coding schemes such as JPEG or JPEG2000. As expected, the filling-in of these layers is not a trivial task.

2. Review of methods for data filling

In the general case, the foreground could be filled with stripes of the same color taken from the neighboring existing pixels, thus ensuring the continuity in the horizontal axis; all other missing
areas of the foreground (areas not neighboring with existing parts) could be filled with the same color (such as white). This way predictive and entropy coders, that work in a horizontal tiling manner and use a horizontally causal neighborhood as their prediction domain, would be matched and would efficiently make use of local statistics, leading to better compression performance. Either way, since the foreground is usually not so populated and dense, compression can be easily optimized and is expected to perform well on the average. But, what really makes the difference in the overall compression of a segmented compound image is the compression of the background layer. This layer has usually much less sparsity and can be filled-in using various methods. In most published works on this subject, two approaches can be identified:

- reconstruction-oriented, and
- strictly compression-oriented

2.1. Reconstruction-oriented methods

In general, in reconstruction-oriented approaches, the filling-in of missing parts is driven by the notion of reconstructing an ideal or fictional model image. The quality improvement after compression is meant here in terms of comparison with the case of compressing without data filling. Many works in other areas of imaging, such as image restoration and reconstruction, deal with this problem under the scope of filling-in or substituting lost and destroyed original image parts (such as in [47]). Other fields could also contribute significantly to reconstruction such as method-specific techniques for removal of compression artifacts, such as in [46]. Nowadays, other promising techniques in the area of reconstruction come from a curators’ point of view and are driven by methodologies for image inpainting, disocclusion and texture synthesis.

In image inpainting methods [1,3,5,33], local characteristics are used in order to propagate information from the existing parts to the missing ones, in an iterative manner. Specifically, by exploiting the local gradient of Laplacian and the isophote direction (orthogonal gradient), image data from existing parts can be iteratively propagated into missing parts. This propagation is achieved by solving the partial differential equation (with $t$ being an artificial time marching parameter)

$$\frac{\partial I}{\partial t} = \nabla (\Delta I) \cdot \nabla I,$$

where $I, \nabla, \Delta, \nabla I$, are the image, the gradient, the Laplacian and the orthogonal-gradient (isophote direction).

Disocclusion [30,29] can actually be defined as a notion opposite to occlusion; occlusion is the effect of an object being in front of another, rendering certain parts of the latter invisible. Reversing this definition one can deduce that the idea of disocclusion is to remove an imaginative object, which would produce a given gap in the background. Generally, disocclusion takes into account level lines in a neighborhood around the missing parts and tries to extend them into the missing areas, while ensuring that several criteria should hold, such as keeping level lines from crossing each other using the definition of T-junctions.

In texture synthesis approaches [19,15,16,45], missing parts of images are totally replaced by specific existing parts according to their neighborhoods. Particularly, in most methods, after defining a neighborhood around the missing parts, a search algorithm locates other similar areas in the image that minimize a distortion measure. When the best match is located, it is used to replace the missing part for which the search was initiated.

Recently, hybrid image-inpainting-texture-synthesis methods [6,36] have also been proposed and their results in reconstructing missing parts of images are quite promising. According to these methods, the missing parts of the image are filled either by inpainting or by texture synthesis according to characteristics of a predefined neighborhood.

Additionally, much work in the area of reconstruction-oriented methods deals with the reconstruction of missing parts after transmission through noisy channels. Especially, in cases where the images were initially encoded in a tiled manner,
error occurrence could result in losing whole tiles. If, for example, JPEG2000 is used, errors could result in either losing whole data blocks or reconstructing erroneous image data. The loss of data blocks is more apparent in JPEG, which is an algorithm specifically designed to work in tiles (the known 8 × 8 blocks). Many works have already addressed the problem of the reconstruction of missing parts of JPEG encoded images that were lost during transmission (as in [39]). In these approaches the decoder tries to reconstruct the missing parts, taking into account the a priori known distributions of DC and AC coefficients [40,37,41] and the characteristics of the encoder.1

2.2. Compression-oriented methods

Compression-oriented approaches of data filling are driven by the notion of improving compression performance without having any visually “pleasing” reconstruction notions in mind.

In [17] Huang et al. proposed a layered method for compressing bank check images where the background data filling is done by replacing masked pixels by the average of their neighboring ones. This method of data filling is obviously not optimal and not matched to any specific encoder, but can produce good results in cases of smooth backgrounds with not very wide gaps.

In [9] Chen et al. proposed an iterative Projections Onto Convex Sets (POCS [2,25,11,10]) algorithm to encode arbitrary shaped image segments using a usual DCT coder. In this work, the sparse image is successively projected from the convex set of images that can be represented by a selected group of transform coefficients to the convex set of images whose pixel values outside the empty areas are specified by the initial image.

The iterative process is guaranteed to converge by the theory of POCS.

Based on this work, Bottou and Pigeon proposed in [8] an extension of the successive projections algorithm into a wavelet coding scheme, where the scale could serve as a significant speed-up factor. This algorithm is actually employed in the DjVu IW44 wavelet encoding engine for the background layer.

In [42] Stasinski and Konrad, propose a variation to [9] where, again, two projection operators are used: bandwidth limitation and sample substitution.

de Queiroz et al. in [14] as well as in [27] and [13] propose a data filling scheme that targets specifically the JPEG encoder using the following data filling approach:

- blocks lying entirely in the background are left intact,
- blocks lying entirely in the foreground are filled with the DC value (mean of pixels) of the previous block (matching baseline JPEG’s DPCM of DC coefficients),
- partially empty blocks are filled by using an iterative approach that propagates values from the existing pixels, using mean values from the immediate neighborhood, thus producing smooth intra-block level transitions, in an attempt to minimize AC coefficients’ energy.

Obviously, the approach was derived with a generic coder in mind, and produces good results, though quite simple. In [13] de Queiroz mentions the possibility to do data filling in the DCT domain (similar to the successive projections algorithm originally proposed in [9]), by also taking quantization into account.

3. Data filling using non-linear projections

Inspired by these works in compression-oriented schemes, we propose a system that actually implements a non-linear successive projections algorithm as described in [9] and briefly presented in [13]. Our purpose was to use a generic compression scheme (JPEG) and test if the

---

1It should be noted that, hereafter, the usage of JPEG implies that the image to be coded is of certain statistics (generally smooth) and meets the encoder characteristics, thus being efficiently compressed. This is true in our case, where the images to be encoded are background layers and are expected to be smooth.
extension of the idea of POCS with the introduction of the non-linearity of the quantizer could actually lead to better compression results.

The baseline JPEG encoder uses DPCM coding to reduce the amplitude of the quantized DC coefficients. At the final stage it employs Huffman coding to encode these DPCM differences as well as the quantized AC coefficients. From this point of view, data filling should be a process of predicting DC values as well as several AC values for each block in order to make the image less demanding of bits during the entropy coding stage.
The overall proposed scheme that was devised to meet these requirements is as follows:

Given an image, a binary mask\(^2\), and the fact that the selected encoder is the baseline JPEG\(^3\), the system:

- performs an initial data filling (prefilling) so as to provide with a good initial condition,
- compresses the prefilled image with JPEG at a given quality, and measures the compression rate,
- decompresses the image, and measures the quality in the originally existing pixels positions (where the mask is false),
- replaces background (initially existing where the mask is false) pixels with their initial values,
- updates the DC values of entirely-empty-blocks,
- iterates until convergence. Convergence in this algorithm is met when no further changes in compression ratio and in quality (PSNR) are measured.

The system can be described by using two sets:

(a) the set of images represented by using a group of transform coefficients quantized by the JPEG quantizer \(q(i,j)\):

\[
Q = \{ \hat{f}(i,j) = f(i,j)/q(i,j), \forall i,j \},
\]

(b) the set of images whose pixel values are equal to the initially existing ones (as defined by the mask):

\[
P = \{ \hat{f}(x,y) = f(x,y), x,y \notin M \},
\]

where \(f\), \(q\), \(M\) are the image, the quantizer, and the mask, \(i, j\) are coordinates in the transform domain and \(x, y\) are coordinates in the spatial domain.

Obviously set (b) is a convex set, while (a) is not. Since this holds, the system cannot be described by the POCS theory and, therefore, cannot be guaranteed to converge to a global minimum. Despite the lack of theoretic proof for convergence, we simulated a system that successively projects between sets \(P\) and \(Q\) (as described above), and found that the system always converges to a sub-optimum. The convergence speed as well as the final approximation of the overall optimum depends a lot on the initial prefilling method used. The proposed system is shown in Fig. 1.

Intuitively, what happens is that during the iterations, information diffuses from the existing to the masked (but prefilled) pixels and backwards. This can be illustrated in a simple example (Fig. 2): in an image stripe of \(8 \times 8\) blocks there are some consecutive blocks that are partially masked, as shown in Fig. 2a, where all missing pixels are.

---

\(^2\)Masks are binary, with true values for foreground and false values for background pixels.

\(^3\)In our tests we have used the standard JPEG coder \texttt{cjpeg v.6b} provided by the \textit{Independent JPEG Group} (\url{http://www.iijg.org}).
drawn with black color. The bounding boxes in this figure highlight the $8 \times 8$ blocks with missing data. At first, all missing values are set to some initial values, using a method from Section 2 (see Fig. 2b). Then, the system iterates until convergence. Fig. 2c depicts the final image when using compression with quality factor 50.

3.1. Prefilling

As we are bound to sub-optimality, it is reasonable to assume that the starting point (or the prefilling method) is a key point to improve the approximation of the overall optimum. To verify this, we have tested a number of prefilling methods.

---

**Fig. 3.** Compression and quality gain of each prefilling method prior to projections compared with no-prefilling prior to projections.

**Fig. 4.** Compression and quality gain by using DC-Huffman instead of no prefilling for images “magazine” and “fay”.
in order to evaluate the effect of each algorithm on the efficiency of the overall system. To this end, the following 13 methods were compared:

(0) no prefilling all masked pixels are black,
(1) prefilling with a mean DC value from the $3 \times 3$ block neighborhood,
(2) prefilling with the DC values’ gradient from existing left and right block neighbors (like fitting a spline using the existing left and right DC values),

(3) prefilling with the weighted mean of DC values taken from the causal block neighborhood (as defined in [39]),
(4) prefilling with the overall image mean,
(5) prefilling with the most probable background color (computed by gaussian smoothing the image histogram until it reaches a bimodal form, and by getting the color that corresponds to the largest peak in histogram),
(6) prefilling with the DC value of the previous block or the weighted mean of DC values of the previous and the current block,

Fig. 5. (a) Test image “fay” and corresponding mask (c), (b) test image “magazine” and corresponding mask (d).
(7) prefilling with the de Queiroz method (as defined in [14]),
(8) prefilling using fast inpainting (kernel 1) (as defined in [33]),
(9) prefilling using fast inpainting (kernel 2) (as defined in [33]),
(10) prefilling using standard inpainting (as defined in [5]),
(11) prefilling using standard successive projections (as defined in [9]),
(12) prefilling with a slightly modified successive projections algorithm that employs wavelets (a one level decomposition similar to the one in [8]).

As expected, method 0 was the worst case scenario, leading to bigger files with lower image quality. Methods 4 and 5 also gave poor results, since they are global methods that fail to capture local characteristics and therefore fail to match the

Fig. 6. Comparison between using no prefilling and prefilling with DC-Huffman: image “ray” at (a) 24967 bytes (32.85 dB PSNR) and (b) 19479 bytes (33.58 dB PSNR), and image “magazine” at (c) 54039 bytes (33.76 dB PSNR), (d) 14500 bytes (36.66 dB PSNR).
encoder. Method 1, even though it is based on DC values' prediction in the DCT domain, did not produce satisfying results as it does not match the encoder's engine. Methods 8-10 based on inpainting are actually methods for reconstruction: their product was an initial image that intuitively resembles the fictional image behind the masked foreground. The results are impressive in terms of reconstruction but the poor compression efficiency as well as the very long computation times required due to the recursive nature of these methods render them inefficient for compression optimization problems. Methods 11 and 12 gave satisfying average results. On the other hand, methods 2, 3, 6 and 7, where the prefilling takes the given encoder into account, gave the best average results. Furthermore, of these methods, the proposed method (6) gave the best average results, both in compression and quality gain. We named this method DC-Huffman since it fills-in with DC values, having the baseline JPEG Huffman coder in mind.

In Fig. 3 we show the overall evaluation of system performance with respect to the prefilling methods for a typical test image [see Fig. 5b] and JPEG quality factor fixed at 50. In the figure, methods are sorted from left to right in order of decreasing compression gain. Improvement is expressed in terms of comparison with the case of using prefill method 0: the compression gain of a method is expressed as the ratio of the file size obtained by the method divided by the file size obtained by method 0, whereas the quality gain of the method is expressed in terms of difference with the image quality obtained, again, by method 0. It should be noted here that all quality measurements are expressed in PSNR terms (in dB); specifically, since color images are involved, a PSNR metric proposed in [28] is used for simplicity, according to which quality is measured in the YUV color space instead of the usual RGB, and the distinct channel measurements are combined using specific weights, following the psychovisual characteristics of the human visual system:

$$\text{PSNR} = \frac{4 \times \text{PSNR}_Y + \text{PSNR}_U + \text{PSNR}_V}{6},$$

(4)

where PSNR$_Y$, PSNR$_U$, and PSNR$_V$ are the corresponding PSNR values of the $Y$, $U$, and $V$ components of the measured images.

The overall compression and quality gains of the proposed method are depicted in Fig. 4 in terms of comparison with the worst case prefilling method 0 for two representative images and for four different quantization tables (JPEG quality factor at 20, 50, 70, and 90). Test image “lay” is an image with a continuous tone multi-color background.

![Fig. 7. Zoomed portion of Fig. 5(a) and (b), respectively.](image-url)
with smooth and highly textured areas, whereas image “magazine” is a standard magazine page scan with a mainly white background. Test images and corresponding masks used in the experimentation are shown in Fig. 5. Notice that the scales in the figures are different to better fit the displayed data.

In Fig. 6 we show the resulting images after projections’ convergence for the case of no pre-filling (method 0) and pre-filling with DC-Huffman (method 6) for the two test images. JPEG quality factor is fixed to 50. If we examine closely the images in Fig. 6a and b (as shown in Fig. 7) it is noticeable that when using method 0 prefilling (a) the final image is not expected to be actually filled with data matching the encoder. Additionally, since sharp edges would be preserved, ringing effects would be significant in the areas around the masked pixels, resulting in further lowering the overall image quality. On the other hand, DC-Huffman prefilling leads to a smooth image without discontinuities or any ringing artifacts. Additionally, the objective (PSNR) as well as the subjective quality of the images that correspond to

![Compression comparison @fixed quality (avg. 38 dB)](image1)

(a)

![Compression comparison @fixed bitrate (avg. 0.34 bpp)](image2)

(b)

Fig. 8. Average compression and quality gain for five data-filling methods: (a) when compressing to a fixed average quality and, (b) when compressing to a fixed average bit-rate.
the non-masked background pixels is significantly better.

For verification purposes, the algorithm was tested on a set of thirteen standard document images, both gray-level and color and turned out to give similar results. During these tests, the Otsu binarization technique [34] was employed in order to produce the binary mask (binarization was performed on the luminance component of the images). The background layer of every image was filled using five methods and compressed with baseline JPEG. In Fig. 8 we show the average compression and quality gain for each of the following data-filling methods:

1. no data filling and application of non-linear projections,
2. filling with background color and application of non-linear projections,
3. DC-Huffman data filling and application of non-linear projections,
4. de Queiroz data filling [14],
5. successive projections [9].

Results in this figure express the performance of each method when compressing either to a fixed average quality (PSNR) (a) or to a fixed average bit-rate (b). As it is apparent, the foreground was not taken into account, since we were interested only in the background layer compression. The overall average result shows that the proposed method (DC-Huffman) leads to better compression results with enhanced quality. The overall average number of iterations needed for the non-linear projections case 3 during these tests was 22, which was the lowest for all non-linear projections methods (1 3) and similar to those that methods 4 and 5 needed. Additionally, in this case, the number of iterations demonstrated an increasing tendency with increasing bit-rate, as shown in Fig. 9, where the average number of iterations is shown as a function of the JPEG quality factor. Specifically, this increasing tendency can be approximated by a fitted line with a slope of about 0.1533, as calculated by applying linear regression (qf being the JPEG quality factor):

\[
\text{iter} = 0.1533qf + 2.6667 = \frac{9}{60} qf + \frac{3}{2}.
\]  

(5)

It should be noted here, that methods that employ inpainting were not tested in these experimentations because, due to their recursive nature (iterative information diffusion), they usually require thousands of iterations and consume significant amounts of computation time. Thus, a comparison with these methods would be "unfair" and without practical importance. As
Fig. 10. Overall average rate-quality measurements for representative JPEG quality factors set to: (a) 90, (b) 70 and (c) 20.

described in the introductory sections, the purpose of using inpainting is on a completely different basis and targets a “perceptual” image reconstruction without any compression considerations in mind.

To overcome the effect of inappropriate segmentation during the binary mask construction by the Otsu method, another set of tests was designed using artificially constructed image masks that mainly captured text and lines. In these tests we used, again, the five aforementioned methods, on the whole set of test images. We compressed using the complete range of JPEG quality factors from 10 to 90 in steps of 10. The following interesting bitrate-quality results were obtained (as shown in Figs. 10 and 11 \(^4\) and summarized in Table 1): while the lowest bitrate was achieved by method 4, the corresponding quality was of the order of magnitude achieved by method 2. The second best bitrate was achieved by the proposed method (3), which gave the best overall average quality measurement. At the same time, the complexities of the different methods led to varying convergence times, showing another advantage of the

\(^4\)It should be noted that in these figures the methods are rearranged and sorted according to their order of achieved bitrate. Given the experiment’s definition, the order of methods displayed in these figures becomes 4,3,5,2,1 in Fig. 11 or the reverse in Fig. 10.
proposed method, which required about 50s on
the average, while methods 4 and 5 required much
more processing time.⁵

These results verify for another time that at the
heart of a complete multi-layered image compres-
sion system should be a very efficient segmentation
technique. The efficiency of such a segmentation
method should be evaluated here in terms of how
much the produced separate layers match the
characteristics of given encoders. It should also be
noted, that the analysis as well as experimentations
on the topic of segmentation are beyond the scope
of this paper. Our purpose on experimenting on
these issues was simply to give an indication of
how much a right or a wrong segmentation can
affect the overall result of compression (when
using a given compression method that has to be
matched).

⁵The current test platform is based upon a MatLab
implementation on a Pentium IV (2.4 GHz) PC running
Microsoft Windows, and cannot be considered as optimized.
Under that scope, the processing times reported here are
supposed to be taken only on a comparison basis.

![Convergence Times](image)

Fig. 11. Convergence times for the different compression methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Bitrate (bpp)</th>
<th>Quality (dB)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4) de Queiroz</td>
<td>0.66</td>
<td>36.5</td>
<td>437</td>
</tr>
<tr>
<td>(3) DC-Huffman</td>
<td>0.72</td>
<td>37.7</td>
<td>50</td>
</tr>
<tr>
<td>(5) Successive projections</td>
<td>0.76</td>
<td>37.6</td>
<td>248</td>
</tr>
<tr>
<td>(2) Background</td>
<td>0.81</td>
<td>36.7</td>
<td>8</td>
</tr>
<tr>
<td>(1) No Prefill</td>
<td>1.27</td>
<td>34.2</td>
<td>67</td>
</tr>
</tbody>
</table>

4. Conclusions

In the fast growing world of digital compound
or documents images, efficient compression can be
achieved by using layered schemes. Layers, defined
by following the MRC model, can be coded with
standard encoders and are expected to be of small
amount of bytes and improved quality. An
important aspect of the whole process is the
correct data filling of the sparse layers, as this
would ensure improvement both in compression
and quality. In this work, inspired by the work on
POCS, we proposed a non-linear projections
scheme for data filling that matches the baseline
JPEG coder and produces better compression results and image quality.

Specifically, the images are compressed with high efficiency leading up to a 70% bit-rate reduction against non-layered compression. The non-linear projections technique had shown to be highly dependent on the initial data filling, and several prefilling methods were tested. The proposed DC-Huffman method has led the system to produce better results compared with other methods. Additionally, the achieved image quality shown an improvement of up to 3 dB with significant reduction in ringing artifacts around the areas of discontinuities.

This system is going to be part of a complete multi-layer compression system for compound images. Considerations about this system involve the compression of the foreground layer, as well as the efficient foreground background layer separation. Our current research is targeted upon these areas facing primarily the segmentation, where text identification and extraction plays the most important role in the final binary mask generation.

References


