Lossless compression of layered documents: how to set the "don't care" pixels

ANNA ANSALONE and BRUNO CARPENTIERI
Dipartimento di Informatica ed Applicazioni "R. M. Capocelli"
Università di Salerno
Fisciano (SA)
ITALY
{aa, bc}@dia.unisa.it

Abstract: - Many bitmaps documents are a superposition of layers with pictures and text.
Simard et al. in [1] have shown that the compression performance on these documents can be improved by separating images from text and by applying different compression algorithms on the two layers. In doing this the pixels not belonging to the layer that is currently being compressed can be treated as "don't care" pixels and they can be opportunely set to enhance compression.
In this paper we present an experimental solution to set these "don't care" pixels and show how, by doing this, we can improve the lossless compression of the image layer.

Key-Words: - Layered documents, text compression, image compression, lossless compression.

1 Introduction

The last twenty years have seen a growing interest in the research and standardization of specific digital compressed formats for the different types of digital information we deal with every day.
There are many standardized algorithms available to compress digital images (JPEG, JPEG2000, JPEG-LS, etc.) and several compression formats for binary text (JBIG, JBIG2, CCIT G4, etc).
Simard et al. in [1] have shown that all these formats fail on bitmaps that contain a mixture of colored images and colored text, for example book covers, catalog pages, flyers, etc.. In fact, when compressing these documents, the user has today only two choices: the first is to improve the compression and quality of the image data by using a JPEG-type algorithm, but this will probably bring an inefficient compression of the text or drawings. The second is to improve the compression of the binary text, for example by using JBIG2, but this will not compress efficiently the image data.
Simard et al. in [1] have proposed to separate text and line drawing from the background image through their SLIm algorithm and then to compress with different algorithms, the two types of data (the acronym SLIm stands for Segmented Layered Image).
SLIm segments the image in three separate components:
1 - Background
2 - Foreground
3 - Binary mask (flags that for every pixel indicate if it belongs to the foreground or to the background).
Text, annotations and drawing are captured through the binary mask and then compressed using an appropriate coder. The rest of the data (image) goes through an image coder:. Every coder shall deal only with the type of data for which it was designed.
It can be shown that this separation improves significantly the compression of the whole document.
After the separation, the pixels that do not belong to the current layer can be treated as "don't care" pixels and they can be opportunely set to enhance the compression of the current layer.
In this paper we present an experimental solution to the setting of these "don't care" pixels and show how, by doing this, we can improve the lossless compression of the picture layer.

2 Problem Formulation

SLIm works in four steps:
- Computation of the mask.
- Segmentation of foreground/ background.
- Codec for the layer related to the images.
- Codec for the mask.

After the mask has been calculated, the segmenter uses the mask to separate the foreground (for example text and/or bitmaps) and the background (image).
They are separately compressed by using off-the-shelf coders to codify the whole bitmap of the background (in which the foreground pixels are substituted by "don't care" pixels) and the whole

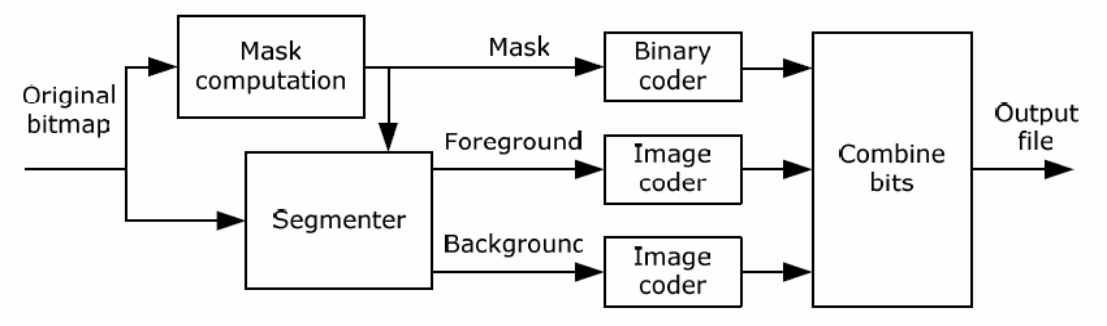


Fig. 1: SLIm

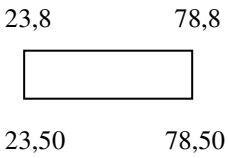
bitmap of the foreground (in which the background pixels are substituted by "don't care" pixels)
SLIm improve the compression of each layer by assigning, to the "don't care" pixels, values that enhance the compression performance of the compressor. Finally SLIm recombines them as specified by the mask.
Figure Fig. 1 (from [1]) shows the SLIm encoder.

3 Setting the "don't care" pixels

One of the critical steps of this compression algorithm is to decide the values that have to be assigned to the "don't care" pixels.
The problem, as stated in [1], is to interpolate the valid pixels in the masked locations in a way that produces a smooth image, without sharp edges. This shall improve compression.
Of course this can be done in many ways, and in our opinion the optimal way to do it depends highly on the type of compression algorithm that will be used to compress the layer.
One of the solutions proposed by Simard et al. in [1] is to run two averaging filters, one that scans the image from left to right and from top to bottom and replace each masked pixel by the average of the left and above pixel, and the other that runs in the opposite direction, from the bottom right of the image.
After the two filters are executed a linear combination of the results from both filters is weighted by the distance of the nearest non masked pixel encountered by the filters and is substituted in the image to the "don't care" values (see Fig. 2 for an example of the result obtained).
Other simple solutions are to assign a constant value (for example 0 or 1) to the "don't care" pixels.

4 Proposed solution

In [1] it is noticed that, for the documents we are dealing with, generally the letters in the text have a constant color.
If we want to compress the image layer we have to smooth the image by opportunely setting the values of the "don't care" pixels of the text layer.
A key observation is that these "don't care" pixels are distributed in the image in regions, corresponding to the text areas, and we can, with good approximation, superimpose to each of these regions simple geometric figures (as rectangles, squares, etc.).
Therefore we devote our attention to the pixels that surround the region. If we know the values of these pixels we can compute the average of this "perimeter".
Therefore, we will assign to the "don't care" pixels this perimeter average, weighted by the original value of the pixel.
In other words:
1 - We calculate the average value of the pixels in the perimeter of the area that contains the text (here for convenience we call it "middle perimeter"). For example if the rectangle whose angles have the followings coordinates:



can be superimposed to the "don't care" pixels of the text layer, we can calculate the average value of the pixels along the perimeter of this rectangle and we call this value middle perimeter.
2 – For each "don't care" pixel we replace its value with the average value between the original pixel value and the middle perimeter of the region to which the pixel belongs.

5 Experimental results

In this paper we are mainly interested in the lossless compression performance of our approach, therefore we have experimented on a set of ten images by using as compression benchmark a standard JPEG-LS

coder (at the moment of writing the coder we have used is available at http://www.hpl.hp.com/loco).
JPEG-LS is today the state of the art of lossless image compression (see [2])
Table 1 summarized the experimental results obtained.
For each of the ten images in the test data set we have reported in Table 1 the compression results obtained by JPEG-LS on the original image (second column), on the image obtained by setting the text layer to a constant value (third and fourth column – see Fig 3 for an example of the original and the modified image). The fifth column shows the compression results obtained by applying the algorithm described in [1].
The sixth column shows the compression obtained on an image in which the text layer has been completely cut, obtaining an image that is smaller than the original one (see Fig. 4).
The last column shows the compression results obtained by our approach (see Fig. 5).
It is important to notice that, in average, our solution improves the compressibility of the image layer with respect to all the other tested approaches.

5 Conclusion

We have presented an approach to the setting of don't care pixels for the compression of composite documents in which we have text superimposed to images.
The solution we presented has been experimentally tested and in our preliminary experiments this solution outperforms the existing strategies in the case of lossless compression of the image layer.
Future research involves deeper experimental testing of the strategy, also in the case of lossy compression.

References:

[1] P. Y. Simard, H. S. Malvar, J. Rinker and E. Renshaw, "A Foreground/Background Separation Algorithm for Image Compression", Proceedings of the Data Compression Conference (DCC'04), IEEE Press, 2004
[2] B. Carpentieri, M. J. Weinberger and G. Seroussi, "Lossless Compression of Continous-Tone Images", Proc. IEEE, vol 88, Nov. 2000

Image	*Original*	*0*	*1*	*Algorithm in [1]*	*Cut*	*Our solution*
Uno.ppm	**3.240:1**	**2.788:1**	**2.799:1**	**2.699:1**	**2.593:1**	**2.433:1**
Diciassette.ppm	**2.361:1**	**2.082:1**	**2.065:1**	**1.873:1**	**1.830:1**	**1.712:1**
Diciotto.ppm	**2.569:1**	**2.101:1**	**2.108:1**	**1.983:1**	**2.003:1**	**1.877:1**
Dieci.ppm	**4.953:1**	**4.065:1**	**4.019:1**	**3.721:1**	**3.981:1**	**3.585:1**
Dodici.ppm	**2.592:1**	**2.213:1**	**2.218:1**	**3.463:1**	**1.967:1**	**1.984:1**
Due.ppm	**3.720:1**	**3.479:1**	**3.475:1**	**2.829:1**	**2.859:1**	**2.374:1**
Island.ppm	**2.551:1**	**2.520:1**	**2.545:1**	**2.090:1**	**2.035:1**	**1.689:1**
Quattro.ppm	**2.740:1**	**2.210:1**	**2.197:1**	**2.049:1**	**2.044:1**	**1.990:1**
Quindici.ppm	**2.818:1**	**2.411:1**	**2.395:1**	**2.224:1**	**2.301:1**	**2.098:1**
Tre.ppm	**2.215:1**	**1.710:1**	**1.721:1**	**1.825:1**	**1.601:1**	**1.691:1**
AVERAGE	**2.976:1**	**2.558:1**	**2.554:1**	**2.476:1**	**2.321:1**	**2.143:1**

Table 1: Experimental results.

Fig. 2: Setting the don't care pixels as in [1].

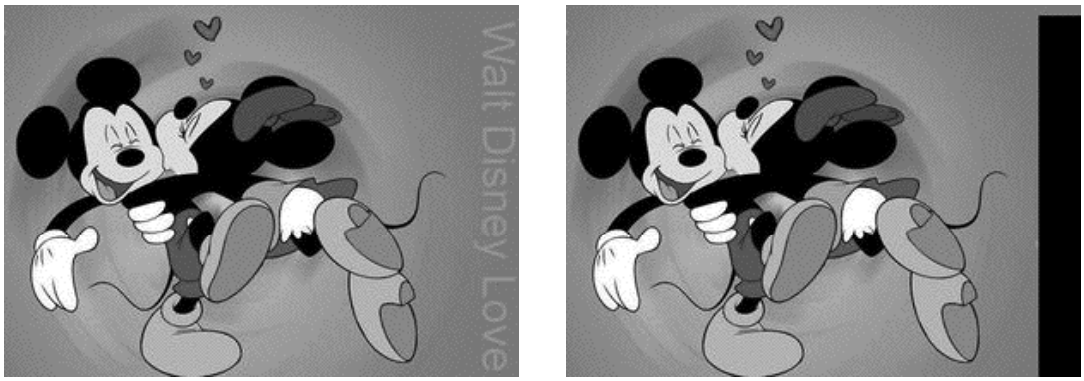


Fig. 3: Setting the don't care pixels to a constant (0).

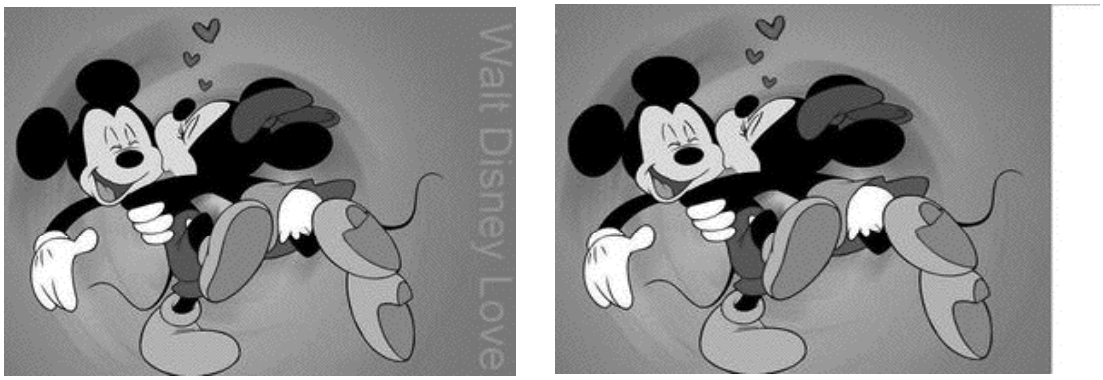


Fig. 4: Cutting the don't care pixels.

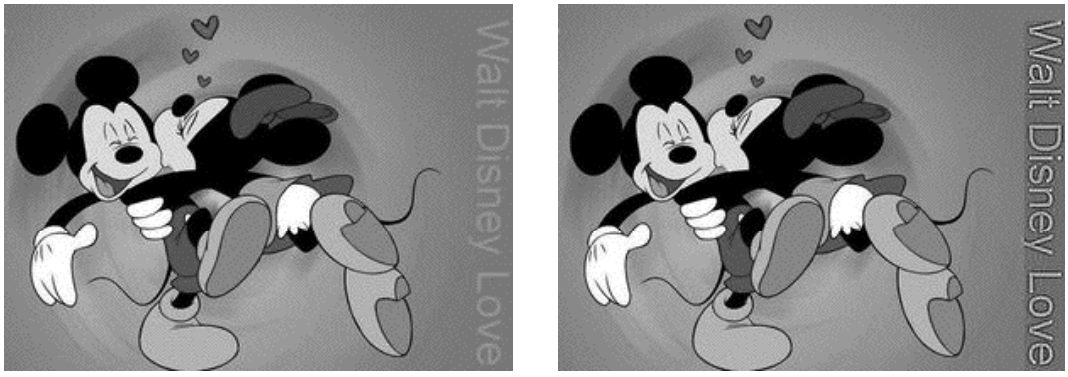


Fig 5: Our solution.