Generating Paper Texture of Historical Documents Using Statistical Moments

C.A.B.Mello & R.D.Lins
Departamento de Informática - Universidade Federal de Pernambuco
Recife - PE - Brazil
{cabm, rdl}@di.ufpe.br

Abstract

This paper presents a scheme for generating paper texture of historical documents. A new entropy based segmentation algorithm is used to decompose the image of documents into the image of the paper background and the printing of the document. Statistical analysis allows filling in the gaps from the printing, yielding a blank sheet of paper with similar texture to the original document.

1. Introduction

Texture, although not formally defined, has its major application on 3D computer graphics to give a more natural look to artificial models. In this paper, we analyse an algorithm to generate paper texture. The algorithm presented was tested on documents from Joaquim Nabuco[1]'s bequest, held by the Joaquim Nabuco Foundation (a social science research institute in Recife-Brazil). The complete file consists of 6,500 documents and about 30,000 pages [3,9]. Nabuco's documents are digitized with 200 dpi resolution, in *true color* (24 bits) and stored in JPEG file format [8] with 1% loss. Figure 1 zooms into a part of an original document with no loss. A segmentation algorithm developed within this project eliminates the ink pixels from an image of a document creating a new image. Our aim is to fill in the generated "holes" yielding the natural look of an aged blank sheet of paper. Statistical measures are used to form the final texture.

The ultimate goal of this work is to be able to regenerate the original document by superimposing the generated writing on a sheet of paper. This is particularly advantageous in the case of typed documents in which the same font-set was used, allowing a very high compression rate in which the image is replaced by text.

The different paper textures can be organised on a database, from which documents share the background texture. Preliminary experiments show that documents can be compressed to about 10% of the original size, being visually similar to the original one. Besides that, our technique makes possible document search and navigation.

[1] Brazilian statesman, writer, and diplomat, one of the key figures in the campaign for freeing black slaves in Brazil, Brazilian ambassador to London (b.1861-d.1910)

2. Segmentation Algorithm

Segmentation objective is twofold [6]: document segmentation and text segmentation. Both works similarly but text segmentation works in a smaller area than document segmentation looking for single characters.

The segmentation algorithm presented in [5] scans the image in search for its most frequent colour. It is likely that this frequency belongs to the background of the image (the paper), as empirical evidence [3] shows that about only 5% of pixels correspond to typed and handwritten parts of documents. This frequency is used as a threshold value to calculate the entropy [1] of black (Hb) and white (Hw) pixels:

$$Hb = -\sum_{i=0}^{t} p[i]\log(p[i]) \qquad Hw = -\sum_{i=t+1}^{255} p[i]\log(p[i])$$

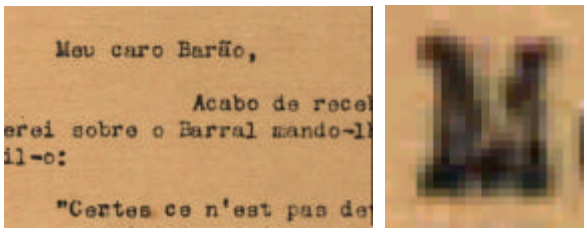

Figure 1. Left) Original image Right) Image detail

where, *p[i]* is the probability of the frequency *i* in the histogram and the logarithm is taken in base 256. The entropy of the complete histogram, *H*, is also evaluated. This value will define two multiplicative factors, *mw* e *mb*, experimentally determined by the following rules:

- If `0.25 < H < 0.30`, then `mw = 1` and `mb = 2.6`
- If `H ≤ 0.25`, then `mw = 2` and `mb = 3`
- If `0.30 ≤ H < 0.305`, then `mw = 1` and `mb = 2`
- If `H ≥ 0.305`, then `mw = mb = 0.8`.

The image is re-scanned with pixel *i* converted to *white* if:

$$(color[i]/256) \geq (mw.Hw + mb.Hb)$$

If this condition is not reached, the color of the pixel remains the same (generating a new true color image) or it is converted to *black* (to generate a monochromatic image).

An example of the use of the algorithm in an image of Nabuco's bequest is found in figure 2.

The segmentation algorithm above was tested on a group of fifty greyscale images from Nabuco's bequest achieving satisfactory results, specially when used on documents written on both sides in which the typing on one side interferes with the other. Amongst the segmentation techniques tested [5], our algorithm yielded the best results, mainly with non-specialized operators.

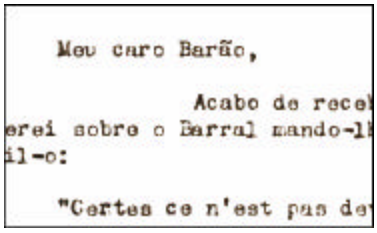

Figure 2. Segmentation algorithm applied to sample document shown in figure 1

The inversion of the condition presented:

$$(color[i]/256) \leq (mw.Hw + mb.Hb)$$

implies that the color of the pixel is turned white, eliminating the frequencies classified as ink (see figure 3). To generate the texture of the paper we work with this variation of the presented algorithm.

Statistical methods are used to fill in these "holes" (white pixels) with frequencies calculated to mantain the aspect of the paper and its real nature.

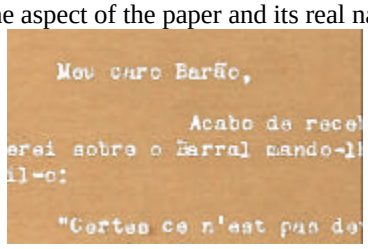

Figure 3. Image of figure 1 segmented to take off the pixels classified as ink

3. True Color Texture Generation

There are several methods to generate patterns of textures [8]. We are interested here in the filling of the white spaces (once occupied by the ink of the documents) with an automatically generated pattern matched with the paper.

A *statistical moment* [6] of order *n* can be defined as:

$$M_n = \frac{1}{N}\sum (x - \bar{x})^n$$

where N is the number of data points, n is the order of the moment and $\bar{x}$ is the average. The mean is related to the first moment (n = 1), and the variance depends on the second moment (n = 2). Other moments can also be used to characterize textured areas. The third and fourth order moments define, respectively, the *skewness* (Sk) and the *kurtosis* (Kurt) [6]:

$$Sk = M_3.s^{\,3} = \frac{1}{N}\sum [(x - \bar{x}).s\,]^3$$

$$Kurt = M_4.s^{\,4} - 3 = \frac{1}{N}\sum [(x - \bar{x}).s\,]^4 - 3$$

Where $\bar{x}$ is the average and σ is the standard deviation calculated over all the non-white pixels in the region.

Figure 4.left zooms into a background image after segmentation (with the ink in white). Figure 4.right shows a straightforward substitution of the white pixels by some color (as the average one). This does not produce a satisfactory pattern. Another problem that can be observed are the borders of letters produced by the *Gibbs phenomenon* [2].

In the first step, we create a greyscale image of the paper. With some information of the original true color image, this greyscaled texture will generate the colored one. Each pixel in a true color image has its color defined by the triple: R (for the *Red* frequencies), G (for the *Green* frequencies) and B (for the *Blue* frequencies).

Figure 4. Left) Background image Right) White pixels colored with average value

The true color input image is converted to greyscale. The greyscale version of the document shown in figure 1 can be seen in figure 5 next.

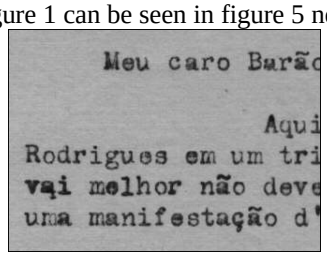

Figure 5. Greyscale version of document of figure 1

3.1 The Proposed Schemes to Create Greyscale Textures

In fact, we show here one algorithm with two possible variations on the fill in process. All the statistical measures can be taken for the complete image or for

parts of it repeating the calculation at each number *d* of lines, 1 ≤ d ≤ y, where *y* is the height of the image. We will call a *region* the number of lines in an image that is being studied (this means, a region can be all the image or a group of *d* lines).

At first, the frequency distribution of the histogram of the image is calculated. In an image with dimensions *x.y*, this distribution is considered uniform and it is calculated by the classical definition of probability:

$$p(i) = \frac{n_i}{n}$$

where *p(i)* is the probability of the color *i* in the histogram (0 ≤ i ≤ 255), n_i is the number of pixels colored with the color *i* and *n* is the total number of pixels. It must be observed that *n* is equal to the *x.d.y* (again, *x* and *y* are the dimensions of the image and *d* is the number of lines of the region) decreased by the number of white pixels found in the image. In the original image of any document tested it was not found the presence of white pixels nor in the frequencies classified as paper nor in the ones that are ink. So all white pixels found now are the "holes"' from the segmentation process. They are the ones previously defined as ink. Next, the average and the variation of the region are calculated.

In the histogram of a region let *mean* be its average value; σ, its standard deviation and *max* is the most frequent color of it. For each region, let *corpix[i]* be the color of the pixel *i*.

Whenever a white pixel is found (that is, when pixel *i* has corpix[i] = 255), the system goes back to its last three neighbours and proceeds with an adjust in their frequencies supposing that they were affected by Gibbs phenomenom. In previous experiments we have found this as an acceptable number of pixels affected by the phenomenom. This adjustment follows the rules:

- if `corpix[i] > (max - σ)` or `corpix[i] < (max + σ)` then `corpix[i] = corpix[i] + 0.2*corpix[i]`
- else, if `corpix[i] ≥ (max + σ)` then `corpix[i] = (mean +- σ/2)`, if i = 0, or `corpix[i] = corpix[i - 1]`, if i ≠ 0

The number of neighbour pixels and the multiplicative value of 0.2 in the top rule above were found experimentally. In the bottom rule, the + or - signal is chosen randomically. An example of this tunning in a true color image can be seen in figure 7.

The system proceeds with the substitution of the white pixel by a color that keeps the original pattern, as much as possible. Three procedures were tested and are analysed next.

Figure 7. Left) Original image segmented Right) Filtering of the image to reduce the effect of the Gibbs phenomenom

3.1.1 Skewness-based Procedure

Whenever the scanning of the region detects a white pixel, the system calculates the new value of the pixel so that it does not change the value of the skewness of the region, using the expression for the evaluation of the *skewness* (Sk) of a region. This means that the pixel will have its color *C* determined by:

$$C = (Sk.N)^{1/3}.s + \bar{x}$$

If *C = 255* (the white color), its value will be changed by the average of the histogram of the region. After the definition of the new color of the pixel (the white color will not be accepted and will be changed to the average value) a new value of the standard deviation will be evaluated adding the new pixel with color *C*. Figure 7 shows the application of this procedure to the image of figure 5 with the use of different values of regions.

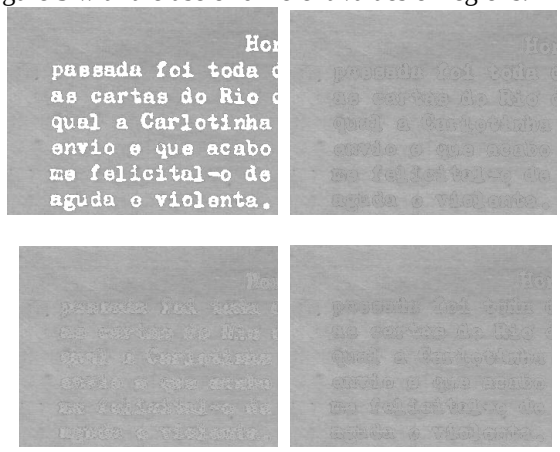

Figure 7. Skewness texture generation method (top left) original image; method applied with (top right) d = y (complete image), (bottom left) d = 1 and (bottom right) d = 10

3.1.2 Kurtosis-based Procedure

This procedure is very similar to the one presented before, but uses now the *kurtosis* measure instead of the skewness. Here, the white pixel will receive a *C* color defined by:

$$C = [(kurt + 3).N]^{1/4}.s + \bar{x}$$

The same restriction to the white color presented before is valid here just as the new evaluation of the

standard deviation after the choice of the *C* value. The use of this procedure in the sample image with several values of *d* is shown in figure 8.

The *skewness* and the *kurtosis* procedures have produced good results and any of them could be used to create the greyscale texture of the paper.

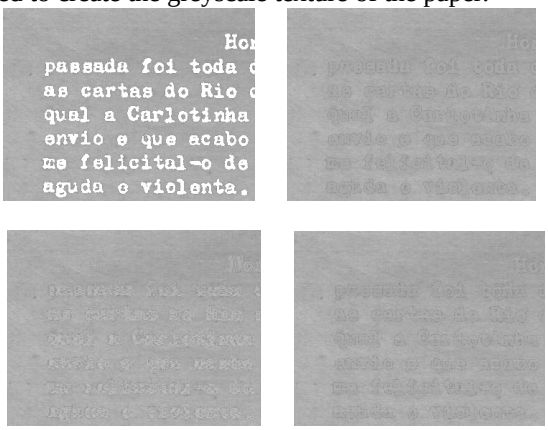

Figure 8. Kurtosis texture generation method (top left) original image; method applied with (top right) d = y (complete image), (bottom left) d = 1 and (bottom right) d = 10

3.2 Adding Color

In the final step of our texture generation scheme, we will colorize the greyscale texture. It is necessary to have some information about the original image (currently, this information are being stored in a separate file). It is stored the average of red, green and blue components of all pixels (except the white ones) and the average value of pixels of the greyscale image (the one going to be colorized). These variables are called *avg_red*, *avg_green*, *avg_blue* (for the true color images average components red, green and blue respectively) and *avg_grey* (for the greyscale average one).

Then, each entry of the greyscale image palette (called *pal*) is compared to *avg_grey* generating a new component of red, green and blue. For the red components this comparison follows the rules:

- if `(pal ≥ avg_grey) then new_color(red) = avg_red - avg_grey + pal;`
- else `new_color(red) = avg_red - pal + avg_grey`

where *new_color(red)* corresponds to the new frequency for the red component of the color. The same comparison is done for the other components (green and blue). All of the three comparisons are used to create a new entry of the new palette.

After the colorizing process, the system proceeds with an application of a suitable low-pass filter to atenuate the high frequencies. Figure 9 shows the use of this colorizing method in the generated texture of figure 7 bottom left. The best quality textures were generated with high values of region (in the examples, d = y and d = 10).

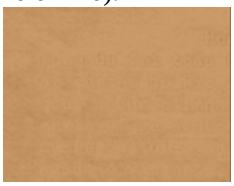

Figure 9. Colorizing the greyscale image of figure 7 bottom left and generating the final texture

4. Conclusions

The texture generating scheme presented herein is part of a compression scheme where a text file will create a perceptually similar version of the original image. A clustering of textures is going to group sets of documents with resembling characteristics. This clustering will store only greyscale images which will be colorized by the system presented here.

As the filling algorithm tries to keep the statistical moments of the image constant the generated pattern is very close to the original. Experiments on document sysnthesis adding the texture produced to the ink image gave very satisfactory results. In special, the generation of a group of documents written on similar paper using the same paper texture yielded very promising results.

Other algorithms for texture generation are being analysed to achieve even better results. The textures database for the complete set of documents in Nabuco's bequest of documents is also being created.

5. References

[1] N.Abramson. Information Theory and Coding. McGraw-Hill Book Company, 1963.

[2] R. Gonzalez and P. Wintz. Digital Image Processing. Addison Wesley, 1987.

[3] R.D. Lins et al. An Environment for Processing Images of Historical Documents. Microprocessing & Microprogramming, pp.111-121, North-Holland, 1995.

[4] C.A.B.Mello & R.D.Lins. A Comparative Study on OCR Tools. Vision Interface 99, pp. 224-232, Quebéc, Canada, May, 1999.

[5] C.A.B.Mello & R.D.Lins. Image Segmentation of Historical Documents (in portuguese). Proceedings of XVII SBT, pp 700-794, Brazil, September, 1999.

[6] J.R.Parker. Algorithm for Image Processing and Computer Vision. John Wiley and Sons, 1997.

[7] K.Sayood. Introduction to Data Compression. Morgan Kauffman Publishers, Inc., 1996.

[8] A.Watt. 3D Computer Graphics. Addison-Wesley Publishing Company, 1995.

[9] Nabuco. URL: http://www.di.ufpe.br/~nabuco.