Open access • Journal Article • DOI:10.1109/TPAMI.2013.63

FAIR: A Fast Algorithm for Document Image Restoration — Source link

Thibault Lelore, Frédéric Bouchara

Institutions: University of the South, Toulon-Var

Published on: 01 Aug 2013 - IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE Computer Society)

Topics: Image restoration, Algorithm design, Image segmentation, Edge detection and Background noise

FAIR: A Fast Algorithm for document Image Restoration

Thibault Lelore, Frédéric Bouchara

FAIR: a Fast Algorithm for document Image Restoration

Thibault Lelore, Frédéric Bouchara

***Abstract*—We present in this paper the FAIR algorithm: a Fast Algorithm for document Image Restoration. This algorithm has been submitted to different contests where it showed good performance in comparison to the state of the art. In addition, this method is scale invariant and fast enough to be used in real-time applications. The method is based on a double-threshold edge detection approach which makes it possible to detect small details while remaining robust against noise. The performance of the proposition is evaluated on several types of degraded document images where considerable background noise or variation in contrast and illumination exist.**

***Index Terms*—Image enhancement, Image edge detection, Image segmentation, Image processing, Image restoration**

I. Introduction

Binarization is one of the initial steps in most document image analyses and understanding systems (initial classification, Optical Character Recognition, etc.). It plays a key role in document processing since its performance affects quite critically the degree of success in a subsequent character segmentation and recognition. Degradations appear frequently and can be due to several reasons which range from the acquisition source type to environmental conditions.

Numerous document image binarization methods try to find a threshold (local or global) which separates text and background thanks to a statistical criterion. Global methods assume that the statistical distribution of background and foreground color is constant for the entire image. An example of such an approach was proposed by Otsu [1]. In this algorithm the global threshold is found by maximizing the separability of the classes. Unfortunately, this approach doesn't work when documents contain stamps or a high range of contrast.

In local methods the threshold is adjusted using local computation. Bernsen proposed a method based on the minimal and the maximal values of a local window [2]. Other works used the standard deviation and mean values [3], [4] or a local contrast thresholding [**?**]. Gatos proposed a method in two main steps: the gray levels of the background are first computed thanks to Sauvola's algorithm and used to binarize the image efficiently [5]. Recently, a new method combined different thresholding methods and then applied a classifier on binarization results to iteratively classify the uncertain pixels as foreground or background [7]. Other works extend the local thresholding approach by integrating spatial information into a Bayesian framework thanks to a Markov Random Field model [8], [9]. Howe proposed to add an automatic parameter tuning to the model using a *binarization instability* measurement [10]. However, these approaches lead to a time consuming minimization of a complex energy function which is incompatible with real-time processing.

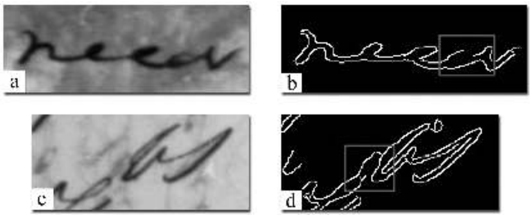

Fig. 1. Illustration of classical drawbacks of Canny edge detector [17]. (a,c) Original documents, (b) No closed boundary and (d) spurious edge responses.

Another approach has the binarization preceded by an edge detection preprocessing step [11]–[16]. This approach is motivated by two main assumptions: text and background are supposed to be mixed in a binary way which leads to sharp transitions. In addition, background artifacts or variations (due to shadows for instance) are supposed to be smooth and have little response to edge detection operators (which is usually the case). This assumption is for instance particularly relevant in the case of bleed-through, due to seeping of ink from the reverse side, or show-through. In addition, edge-based algorithms generally have the advantage of being insensitive to the size of the characters. Indeed, from a given size, the edges of objects are detected regardless of their size.

However, edge detectors suffer from the classical drawback of disjointed contours (figure 1(a,b)) which is a critical problem in the case of documents binarization.

To solve this problem, Cao et al. [11] proposed a modified version of the Canny algorithm which aims at closing the letters's boundary by using the orientation computed at the end of the edge. In a recent work, Chen et al. [12] proposed an improvement of this approach by adding to edge orientation some information about the distance from nearby edges. The resulting contour is then used to estimate the values of two predefined thresholds. The result of high thresholding is used as the initialization step of a region growing algorithm. The final result is obtained by combining the previous result with the low threshold binary image.

With a similar approach Li et al. [13] proposed a method based on the Laplacian operator to select pixels involved in the computation of the global threshold. In [14], Ramirez et al. introduced the notion of *transition pixels* to represent pixels close to edges. From this transition set they compute

T. Lelore and F. Bouchara are with the LSIS laboratory, Southern University of Toulon-Var, BP 20132, 83957 La Garde Cedex, France. E-mail: thibault.lelore@gmail.com, bouchara@univ-tln.fr

a gray-intensity threshold using classical statistical methods. Block and Rojas [15] proposed an optimization for the energy computation of *transition pixels* using the differences between the original document and a blurred version of the image. They use some rules based on the size of connected components to remove the misclassified objects. In the same spirit, Fabrizio et al. [16] search for regions having minimal contrast[1] and use a morphological operator (Toggle Mapping) to find the class. They finally find the class of homogeneous regions using the size and class of these regions' boundaries.

All these methods usually require the estimation of several parameters (such as the size of the analysis window) which have to be adjusted differently for each image. These drawbacks make these algorithms ineffective on documents containing several kinds of text font. Such a problem is particularly significant for the approaches based on a learning process such as MRF models.

With the emergence of handheld multimedia devices (PDA, smartphone, etc.), new applications of these algorithms have also emerged where computation time as well as robustness are crucial. An example of such an application is given by the Google image recognition project *Goggles* [18], which aims at developing searches based on pictures taken by handheld devices. In this paper, we propose an algorithm for the binarization of text-based images. The method can be considered to be parameter-free (as seen in section III-B, the parameters don't greatly affect the quality of the binarization) and, since it is edge-based, scale-independent. It can hence compute binarized images with the same efficiency, whatever the content (font size, variable background intensity, shadows, smear, smudge, low contrast, bleed-through...). The proposed algorithm requires very short computational time without sacrificing performances as indicated by the results of several contests where parts of the algorithm have been submitted:

- A preliminary version of the S-FAIR sub-processing (see section II-A) has been submitted in the DIBCO09 contest [19] and finished 9^{th} out of 43 submitted algorithms.
- The present S-FAIR sub-processing has been submitted to the two following contests:
  - ICFHR 2010 - Quantitative Evaluation of Binarization Algorithms [20] (First out of 6 submitted algorithms for the first type of documents and second out of 6 for the second type of documents.)
  - H-DIBCO10 [21] (3^{rd} out of 17 submitted algorithms.)
- A preliminary version of the FAIR algorithm, which is the subject of the present paper, has finished at the 1^{st} place out of 18 algorithms in the DIBCO11 contest [22] and 2^{nd} place out of 24 in the H-DIBCO12 [23]

The rest of paper is organized as follows. The proposed method is described in the next section. In section III, the performance of this algorithm is assessed and compared with other methods in the literature. Section IV offers the conclusion.

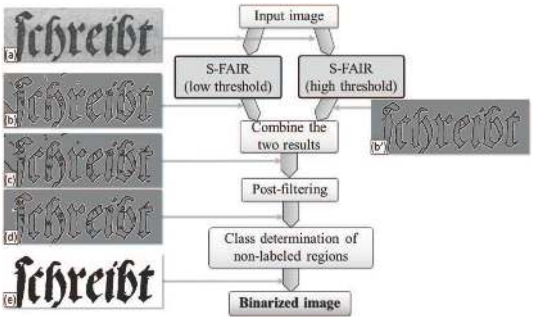

Fig. 2. Block diagram of the proposed algorithm (see section II-B for a complete description).

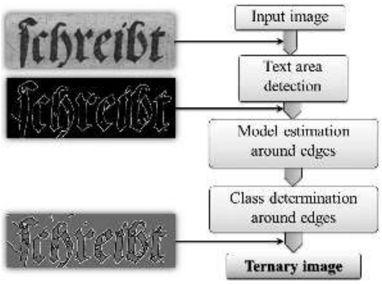

Fig. 3. Details on the sub-process S-FAIR.

II. Proposed method

The proposed method is described in figure 2.

The idea developed in the FAIR algorithm is to combine the results of two different ternary images given by the S-FAIR algorithm (described in the next subsection) using two different thresholds: the first ternary image is considered as noise-free but without some important edges, the other contains each character's edges but with some additional noise. From this merging, we identify problematic areas where some differences occur between the two images. The final binary image is then obtained by recalculating the class of pixels around these problematic areas using a new neighborhood (not the classic square window).

A. The S-FAIR sub-processing

The S-FAIR sub-process (for Simplified FAIR) that we propose is based on a simple algorithm (see figure 3) that can be divided in two main steps as follows:

- In a first step, a rough localization of the text is achieved using an edge-detection algorithm based on a modified version of the well-known Canny method.
- From the previous result, pixels in the immediate vicinity of edges are labeled in in 'text' or 'background' thanks to a clustering algorithm. The remaining pixels (far from an edge pixel) are labeled as 'unknown'. The result is hence a three class image.

Using an additional labeling process described in subsection II-B4 (to remove the "Unknown" label), this algorithm itself becomes a binarization method. The above-mentioned two steps are discussed in the next sub-sections.

[1] For the DIBCO Contest, they use a hysteresis threshold

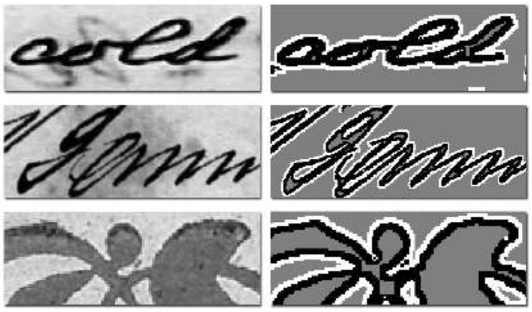

Fig. 4. (left) Original documents, (right) Three values images produced by S-FAIR.

1) Text area detection: In our approach, text localization is based on the well-known Canny algorithm. The given result of this edge detector is strongly conditioned by the tuning of the two parameters T_u and T_l which correspond to the upper and lower thresholds of the hysteresis process.

Applied on a text document with sub-optimal values, Canny algorithm usually leads to several kinds of problems (as illustrated in figure 1). However, it is worth nothing that in our approach, tuning this threshold is not critical. Indeed, edge detection step consist only in a rough localization of the text's position and is followed by a labeling process which enhances the estimation's precision.

As seen in the literature [24], [25], the estimation of T is achieved by applying the Otsu algorithm to the gradient magnitude computed thanks to the Sobel operator. From the threshold T_o computed by the Otsu method, we simply define $T_u = k.T_o$ where k is the only parameter of the sub-process.

The second threshold T_l is usually computed from T_u by a simple linear relation: $T_l = \alpha.T_u$ with α usually chosen in $[0.3, 0.5]$.

2) Model estimation around edges: The second step of the algorithm is devoted to the classification of the pixels close to the edges previously detected. To this aim, we simply define our observational model as a slow varying underlying image disturbed by a centered white Gaussian additive noise, that is:

$$o_i = (1 - z_i).(\mu_i^b + n_i^b) + z_i.(\mu_i^t + n_i^t) \tag{1}$$

In the previous equation, o_i is the observation image at site i. μ_i^b and μ_i^t are the noiseless version of respectively the background and the text. z_i is the hidden binary variable to be estimated with $z_i = 0$ for the background and $z_i = 1$ for the text. We assume that the covariance matrices (Σ^t and Σ^b) of the noise (n_i^t and n_i^b) only depends on the nature (text or background) of the pixel under consideration. The values of μ_i^b and μ_i^t are estimated for each pixels i in the vicinity of the edges along with the values of z_i. Conversely, the noise is assume to be independent of the site i and the values of Σ^t and Σ^b remain constant for all the pixels of the image. Another reason for this choice is to estimate with greater precision and robustness these parameters which are notoriously sensitive to noise.

Let $N(s)$ be an $(n \times n)$ square window centered on edge pixel s (detected in the previous step). Such a window contains both text and background and, thanks to the previous assumption, the statistical distribution of its pixels can be modeled by the mixture of the two d-dimensional Gaussian processes $\mathcal{N}_d(\mu^b, \Sigma^b)$ and $\mathcal{N}_d(\mu^t, \Sigma^t)$ (where $d = 1$ or $d = 3$ for, respectively, a gray level or a color image). To compute the parameters of this model we use the EM algorithm [26], the principle of which we are going to recall in order to introduce our notation.

The EM algorithm is a classical approach to estimate the parameters of a mixture model by iteratively applying the following two steps:

- The **Expectation step (E-step)** computes the expected value, with respect to the conditional distribution of z given o, of the log likelihood function $\log(P(o, z/\Psi_s))$ where Ψ_s is the parameter vector of the model computed at pixel s, $o = \{o_i/i \in N_s\}$ and $z = \{z_i/i \in N_s\}$.
- The **Maximization step (M-step** estimate Ψ_s by maximizing $Q(\Psi_s/\Psi_s^{(q)}) = E_{z/o,\Psi_s^{(q)}}[\log(P(o, z/\Psi_s))]$

In the above, (q) represents the iteration step of the algorithm.

In the case of a Gaussian mixture model, the log likelihood function is given by:

$$L(o, z, \Psi_s) = \sum_{i \in N(s)} \Big(z_i \log F(i, t) + (1 - z_i) \log F(i, b) \Big) \tag{2}$$

with

$$F(i, c) = \pi_s^c.\mathcal{N}_d(o_i; \mu_s^c, \Sigma^c) \tag{3}$$

where (π_s^t, π_s^b) control the "mixing" value between the Gaussians, c is the class (either $'b'$ or $'t'$) and $\Psi_s = (\pi_s^t, \pi_s^b, \mu_s^t, \mu_s^b)$ is the vector of unknown parameters estimated during the M-step.

The E-step estimates the quantity $Q(\Psi_s/\Psi_s^{(q)})$ by computing, thanks to the Bayes theorem, the conditional distribution of the z_i given our current estimate of the parameters Ψ_s^q at step (q):

$$\begin{aligned} t_i^{c(q)} &= P(z_i/o_i, \Psi_s^q) \qquad (4) \\ &= \frac{\pi_s^{c(q)} \mathcal{N}_d(o_i; \mu_s^c, \Sigma^c)}{\pi_s^{t(q)} \mathcal{N}_d(o_i; \mu_s^t, \Sigma^t) + \pi_s^{b(q)} \mathcal{N}_d(o_i; \mu_s^b, \Sigma^b)} \end{aligned}$$

The $Q(\Psi_s/\Psi_s^{(q)})$ quantity is hence given by:

$$Q(\Psi_s/\Psi_s^{(q)}) = \sum_{i \in N(s)} \Big(t_i^{t(q)} \log F(i, t) + t_i^{b(q)} \log F(i, b) \Big) \tag{5}$$

Thanks to the quadratic form of $\log F(i, c)$, the maximization of $Q(\Psi_s/\Psi_s^{(q)})$ with respect to Ψ_s (the M-step) is straightforward and can be achieved separately for the different

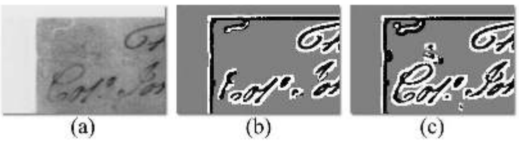

Fig. 5. Example of threshold problems. (a) input image, (b) high threshold misses some edges, (c) low threshold detects noise. We can also notice that sharp transitions in background are detected as text.

terms:

$$\pi_s^{c(q+1)} = \frac{1}{n^2} \sum_{i \in N(s)} t_i^{c(q)} \tag{6}$$

$$\mu_s^{c(q+1)} = \frac{\sum_{i \in N(s)} t_i^{c(q)} o_i}{\sum_{i \in N(s)} t_i^{c(q)}} \tag{7}$$

The initialization of parameter Ψ_s is achieved by using the classical K-Means.

Since the second order properties of the two Gaussian processes are supposed to be constant, $\left(\Sigma^t, \Sigma^b\right)$ constitutes a global parameter of our model and is hence classically estimated at each (q) step using the whole image. Each (i, j) component of the two covariance matrices is hence given by:

$$\Sigma^{c(q)}(i,j) = \frac{\sum_s t_s^{c(q)} \left(o_s(i) - \mu_s^c(i)^{(q)}\right)\left(o_s(j) - \mu_s^c(j)^{(q)}\right)}{\sum_s t_s^{c(q)}} \tag{8}$$

The two steps of the EM algorithm are iteratively applied alternating with the estimation of $\left(\Sigma^t, \Sigma^b\right)$ for each edge pixel until convergence. For the remaining pixels close to edges (*i.e.* belonging to at least one window $N(s)$) we estimate the two values $(\tilde{t}^t, \tilde{t}^b)$ by computing the mean value of the (t^t, t^b):

$$\tilde{t}_i^c = \frac{1}{\#s/N(s) \ni i} \sum_{s/N(s) \ni i} t^c(\Psi_s) \tag{9}$$

where $s \in \{\text{edge}\}$ and $\{\text{edge}\}$ is the set of all edge pixels.

The final class of a pixel i is estimated thanks to the following rule:

$$z_i = \begin{cases} unknown & \text{if} \quad \min_{s \in \{\text{edge}\}} \mathrm{d}(i,s) > n/2 \\ 0 & \text{else if} \quad \tilde{t}_i^{\,t} < \tilde{t}_i^{\,b} \\ 1 & \text{else} \end{cases} \tag{10}$$

where $d(i, s)$ is the city-block distance between pixels i and s, and n is the size of the window $N(s)$.

At the end of this stage, all the pixels of the image close to edges are labeled either 'text' or 'background', the other pixels are temporary labeled as 'unknown' (figure 4).

B. The FAIR algorithm

Before describing FAIR, we shall first clarify some aspects of S-FAIR on which it is based. The results given by S-FAIR presents two main problems. The first one, illustrated by figure 5 (b,c), is a classical problem when using a thresholding approach: there is no optimal threshold value. A high value of k leads to text with open edges. Conversely, a low value of k leads to an accurate detection of the text but at the cost of a significant noise. The second problem of S-FAIR is encountered when the background contains sharp transitions which are then detected as text.

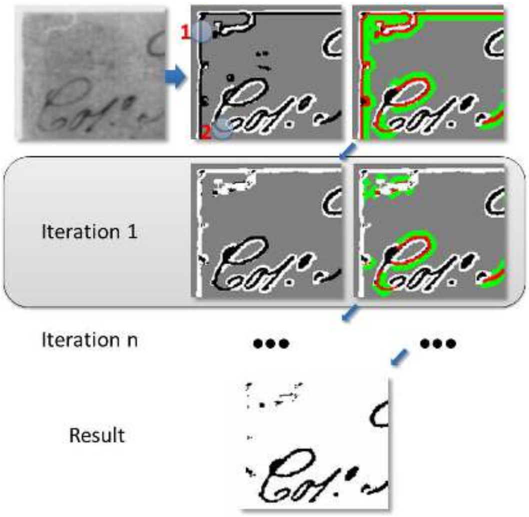

Fig. 6. Detection of misclassified text pixels. First row: (left) Input image, (middle) Three-values image created by the fusion stage, (right) in red, the problematic text pixels (Zp_t^i) and in green, the surrounding area used to compute labels (Zp_b^i). Second row: a new ternary image is produced using $N_f(s)$ and the process reiterate until convergence.

The idea developed in the FAIR algorithm is to combine the results of S-FAIR obtained with two different thresholds (see figure 2d). The image is then filtered in a second step by taking advantage of the particular noise induced by this process. This post-filtering will be described below. As we shall see, the two kinds of misclassified text pixels (noise and sharp transition, see figure 5) share some properties which give us the possibility of removing them using the same process. The global description of FAIR is summarized in figure 2.

1) Double thresholding: To achieve the double thresholding we introduce a new parameter ($K = 1.0$) to adjust the parameter of the two S-FAIR sub-processes (see II-A1). The low threshold k^l is chosen in order to get all the edges of the text. Typically we take $k^l = 1.4 * K$. The high threshold k^h is taken equal to $1.66 * K$ to limit noise influence.

2) Merging: The two previous results are then combined into a ternary image, by respectively associating the numerical values 1, 0.5 and 0 to the symbolic labels 'text', 'unknown' and 'background', and then computing the new ternary image I_m as:

$$I_m = \max(I_{k^l}, I_{k^h})$$

3) Post-filtering process: The approach described in the previous paragraph gives an image with over-detected text labels with many false detections. That's why this post-filtering process considers only the case of misclassified text pixels.

If we consider figure 6(b-c), we can observe that both kinds of misclassified text pixels are neighboring pixels of 'unknown' label (see figure 6 (d)). However, such neighboring pixels are sometimes correctly classified. The post-process has to take this into account to remove only misclassified text pixels.

The filtering is achieved in two steps. The first step removes stains which are detected as connected components only surrounded by 'unknown' pixels (such as the stain above the $\mathscr{C}$ in figure 6(C)).

The second step is iterative and is defined as follows: Let $Z^{t(i)}$ be the set of pixels of 'text' label at the i^{th} iteration. We define $Zp^{t(i)}$ as the set of text pixels which are suspected to be misclassified as follows:

$$Zp^{t(i)} = Z^{t(i)} \cap \mathcal{D}_5(Z^{u(i)}) \tag{11}$$

where $Z^{u(i)}$ is the set of pixels of 'unknown' label at the i^{th} iteration and $\mathcal{D}_5(.)$ is a morphological dilation using a $5 * 5$ diamond-shape structuring element.

At each iteration of the algorithm, a new label for each pixel $s \in Zp^{t(i)}$ is computed. To this aim we define the following neighborhood of s:

$$N_f(s) = N(s) \cap \left(Zp^{t(i)} \cup Zp^{u(i)}\right) \tag{12}$$

where $N(s)$ is the neighborhood defined in section II-A2 and $Zp^{u(i)}$ is computed in the same way as $Zp^{t(i)}$ but this time the text regions grows into the 'unknown' regions. The size of the structuring element used to compute $Zp^{u(i)}$ will be discussed in section II-C. The computation at each stage is achieved thanks to the EM algorithm (eq. 5-10) but using the new neighborhood $N_f(s)$.

This iterative process ends when the labels remain unchanged between two iterations.

To illustrate how this approach works, let's consider the two cases defined by the small circles 1 and 2 of figure 6. In both cases text pixels are neighboring of 'unknown' pixel.

Case 1 corresponds to the detection of spurious text pixels due to a sharp transistion of the background. Thanks to the definition given by eq. 12, the new neighborhood $N_f(s)$ only contains pixels of the dark side of the background. This kind of situation is detected in the clustering step of section II-A2 when the two mean values μ_s^b and μ_s^t are close. The label of pixel s is then defined as 'unknown'.

In case '2' the neighborhood $N_f(s)$ contains both text and background pixels which leads to the detection of two different processes. The final label then remains unchanged.

4) Final labeling: The last step of the algorithm is devoted to the labeling of remaining unknown pixels. Intuitively, it would be natural to consider that an unknown pixel adjacent to a known one gets the same label. This consideration leads to a first naive approach based on the propagation of the label from known pixels thanks to a growing region strategy. However, this method is highly dependent on the initial pixel from which the growing region starts, and may lead to the propagation of misclassification if the label of this pixel is noisy (figure 7). In a recent work, Su et al. proposed an iterative algorithm to classify this type of pixels using neighborhood labels and their corresponding contrast and intensity features [7].

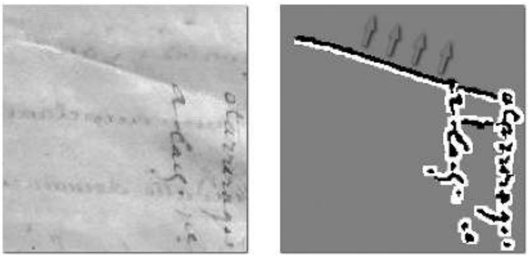

Fig. 7. (left) Original document, (right) Problem to define the class near the black line. The arrows show how the misclassified text pixels will expand.

We have defined a different strategy in two steps: we first extract the connected unknown area and then assign the same label to all the pixels belonging to this area. Extraction of a connected area can be done very quickly by using a connected component algorithm such as the method described in [27]. Although this method is less accurate than Su's [7], this step is both quick and satisfactory in terms of binarization quality.

The estimation of the label assigned to the resulting area depends on the labels of its neighboring pixels, which, by definition, are all known. This estimation is achieved by applying the following rule:

$$z_I = \begin{cases} 1(\text{text}) & \text{if} \quad N^t > \beta.N^b \\ 0(\text{background}) & \text{else} \end{cases} \tag{13}$$

where N^t, N^b are respectively the number of text and background pixels in the boundary of the I^{th} area and β is a weighting used to adjust the sensitivity of the algorithm to the text.

C. Parameters

Although the algorithm presented above includes some free parameters, most of them can be considered as having little or no influence. We will detail here what impact each parameter has on the quality of binarization.

The less important parameter is probably the weighting 'β' of eq.13. Indeed, while this parameter can produce bad results if the value is maladjusted, a unique value works on all test images with near-optimal results (we used $\beta = 1$ in our experiments).

The various neighborhood we use in this method can also be taken as constant with negligible consequences for all images tested, either because they do not strongly influence the result or because the best value appears not to fluctuate much for different images:

- The size of the neighborhood used in II-A2 should be large enough to correctly estimate the model, but small enough not to suffer from the "global thresholding" problem and being fast enough (see Introduction). The experimental results show us that $n = 3$ is a good compromise. Hopefully, this size does not depends on the size of the text nor on the image resolution.

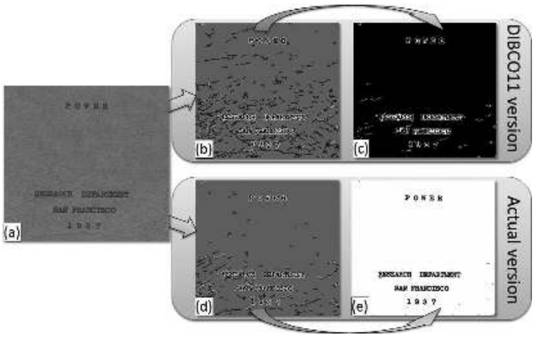

Fig. 8. Text area detection problems. input image (middle), Problematic DIBCO11 version (top), proposed method (bottom). (b) and (d) ternary images, (c) and (e) binarized images.

- The post-processing neighborhood ($Zp^{u(i)}$, see section II-B3) is a little bit more complex: three parameters are involved. Two of them determine the subset of pixels to keep for the model estimation (size and shape of the structuring element) while the latter corresponds to the size of the window. After studying the influence of these parameters, it seems that the shape of the element is the less critical: the choice was focused on a diamond-shape structuring element because it is fast to compute and has good geometric properties. The size of this element is a bit more important because it determines the pixels to keep for the model estimation. A too small value would remove too much pixels from the analysis window, while a too high value could add pixels previously considered as problematic. However, this influence remains low. According to the tests we have achieved, for a width of the structuring element ranging from 5 to 50 pixels, the F-Measure varies from 93.20 ($5*5$) to 93.55 ($29*29$). We have achieved similar assessments on the size of the window $N(s)$ and we have found that changes in F-Measure are less than one percent. For the various tests, we have set this value to ($75*75$) which is a good compromise between computation time and performance.

The parameter which most strongly influences the binarization outcome is the parameter K which controls the text area detection. It is indeed these areas that affect the rest of the algorithm and thus deserve particular attention to prevent bad detection. A particular tuning of this value can give good results for some images and be ineffective for others. Such a problem occurs for instance in the DIBCO11 competition for two images: PR6 and PR7 (see figure 8). Indeed, the detected text areas were too numerous and the algorithm thus labeled the large 'unknown' area as text. This particular problem can be solved in different ways. One can assume that the largest 'unknown' area is necessarily 'background', but this solution involves making assumptions about the layout of the documents. Another solution would be to use the ratio of 'text' labels to 'background' labels to detect problems, but this would imply re-estimating the whole image and thus be more time-consuming. We chose to add the additional step described in section II-B3 which removes the 'text' connected components surrounded by 'unknown' labels. This way, a lot of erroneous labels are removed and the 'unknown' areas are then correctly classified.

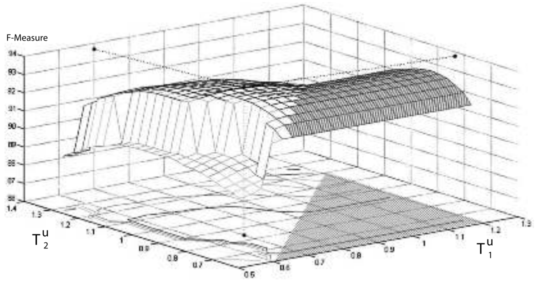

Fig. 9. Evolution of the F-Measures as function of T_u^1 and T_u^2 (means calculated on 54 images). The shaded area corresponds to parameter combinations that avoid the use of double thresholding ($T_u^1 > T_u^2$).

Due to our 'double threshold' approach, parameter K implicitly corresponds to four parameters: low (T_l) and high (T_u) Canny's threshold for both sub-processes. We first decided to study how the low parameter impact binarization output by varying α, the ratio between the two thresholds: $T_l = \alpha.T_u$. We found that the same value works on all test images with near-optimal result when $\alpha = 0.38$. We then studied the impact of the upper threshold of the two sub-processes. As shown by figure 9, the value did not greatly influence the binarization outcome and the best result was obtain with $T_u^1 = 1.4$ and $T_u^2 = 1.66$. We then set k^l and k^h presented in section II-B1 according to these values. In addition, the figure 9 indirectly shows the impact of the double thresholding: the shaded areas represents areas where the merging process is impossible ($T_u^1 > T_u^2$). We hence can see that the F-Measure has increased by approximately 1% thanks to that improvement.

III. Experimental results

This section presents an assessment of the proposed method, based on its comparison with several algorithms from the literature. The retained criteria are first the algorithms' run time, then the results obtained in the binarization of handwritten and typewritten documents through three different methods: the F-measure proposed in the DIBCO09 contest, visual considerations and the recognition rate of an optical character recognition software (ABBYY FineReader 9.0 [28]).

To compare the results of our algorithm, the documents used in our benchmarking are derived from several sources such as the Oulu database[2], the DIBCO contests [19], [21]–[23] or private collections. The implementation of our algorithm don't use the computational extensive version proposed in II-A2. Instead, we use the fast K-Means algorithm to estimate the various values of our model, but you will find a performance comparison between the two implementations in table II.

A. Run time

For the run time comparison, we have chosen five different algorithms from the literature. Our algorithms (k-means and EM version) are compared with three fast and well-known methods: the Otsu method [1] (Chung's optimization [30]),

[2] http://www.mediateam.oulu.fi/MTDB/

TABLE I
COMPARISON OF RUN TIMES (MILLISECONDS/MEGAPIXEL) ON VARIOUS KINDS OF IMAGES.

Image type	Otsu	Shafait	Sauvola	Fabrizio	Ramirez	FAIR	FAIR
						k-means	EM
Handwritten							
Max.	10.4	85.7	10 021.1	1 979.3	2 023.2	755.1	35 789
Mean	6.8	79.3	6 480.6	1 227.3	1 460.1	619.8	30 489
Min.	2.2	65.2	750.0	823.5	1 040.9	578.3	27 890
Typewritten							
Max.	10.6	83.5	7 041.7	2 449.2	2 513.1	725.6	33 485
Mean	9.5	79.5	4 089.0	1 547.5	1 430.8	670.5	31 563
Min.	8.9	75.7	1 252.2	540.7	936.2	547.2	29 990
Natural scenes							
Max.	15.3	84.7	10 722.0	7 038.0	2 744.0	782.4	34 450
Mean	13.0	78.8	4 113.5	3 190.4	1 603.3	575.1	28 044
Min.	9.0	72.9	976.7	1 886.7	1 180.7	517.5	26 672

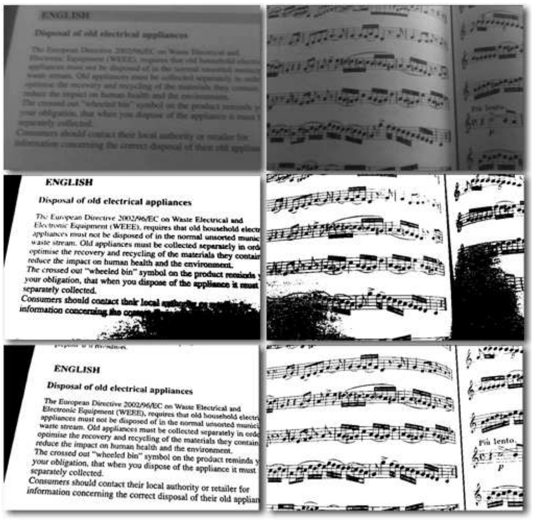

Fig. 10. Binarization of live images using a smartphone. (top) Grey input, (middle) Otsu's method, (bottom) proposed method.

the original [4] and an optimized version [31] of Sauvola's algorithm. We have also compared our algorithms with two recent proposals by using the binary code proposed by their authors: the algorithm of Fabrizio et al. [16] and the method of Ramirez et al. [14]. For all these algorithms we have run the binarization of 27 documents 200 times under the same environment (Intel® Core i7 @ 2.8 GHz, 2GB Memory) and we have implemented the algorithms in C++ except for Fabrizio and Ramirez's methods, for which we used the authors' binary code. The documents are quite large (between 1.5 and 6.9 mega pixels) and taken from the different sources mentioned above. Note that, this EM version is implemented for the general case (i.e. for a color image) and involves matrix calculations which explains the long run times.

The comparison of execution times is summarized in table I. As expected the fastest results are given by the Otsu method which is simply based on a global thresholding approach. Roughly speaking, our proposal is about ten time as slow as Shafait's optimization of the Sauvola algorithm but twice as fast as the algorithms of Ramirez and Fabrizio which have comparable performance.

We can note that there is a large gap between the maximum and minimum values for the original Sauvola's method due to the different window sizes ($13 * 13$ to $50 * 50$). Shafait's optimization makes the run time quite independent on the size of the window and leads to an improvement by a factor of 50. We can also observe similar gaps for Fabrizio's, Ramirez's and our methods but they have different explanations. They are due to the fact that the run time highly depends on the number of detected edges. However, our algorithm remains reasonably sensitive as indicated by the value of the run time variance (16 ms/megapixel for our algorithm, 176 ms/megapixel for Ramirez's algorithm and 320 ms/megapixel for Fabrizio's algorithm).

In order to assess the relevance of this method in a mobile context, we have implemented the k-mean version of the FAIR algorithm on a smartphone (ARM Qualcomm®MSM8660 @ 1.2 GHz, 1GB Memory, resolution: $720 * 480$) to binarize natural scenes taken from the camera. With such equipment, our method processes around 1 frames per second. In comparison the Otsu's method processes 60 frames per second in the same context. However, as shows figure 10 which gives some examples of results, the global thresholding of the Otsu's method is unable to cope with the natural-lighting scenes.

B. Binarization quality

In this section, we compare the binarization quality of our approach with the following five methods: Otsu [1], Sauvola [4], Chen [12], Ramirez [14] and Fabrizio [16]. This comparison is based on the F-measure used in the DIBCO contest (Eq.14) when a ground-truth is available, and when there is none, on visual considerations.

The F-measure (FM) is defined by the following expression:

$$\mathrm{FM} = 100 * \frac{2 * \mathrm{Pr} * \mathrm{Rc}}{\mathrm{Pr} + \mathrm{Rc}} \tag{14}$$

$$\text{with:} \qquad \mathrm{Rc} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \text{and} \qquad \mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

TP, FP, TN and FN denote respectively the true positive, false positive, true negative, and false negative values.

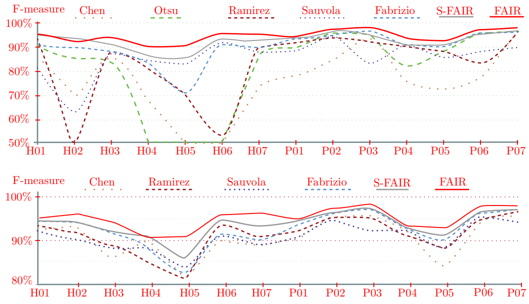

Fig. 11. Performance comparison in terms of F-measure without adjusting parameters (top) and with optimum parameters (bottom).

TABLE II

BINARIZATION EVALUATION USING VARIOUS DATABASES: DIBCO9 (H01-P07) ; H-DIBCO 2010 (H01'-H10') ; DIBCO11 (HW1-PR8) ; H-DIBCO 2012 (H01"-H14")

Method	Expectation-Maximisation				K-Means	
Docs	FM	PSNR	DRD	MPM	FM	PSNR
H01	95.203	21.844	1.278	0.2	95.2005	21.8456
H02	92.529	24.714	3.277	0.101	92.578	24.7445
H03	94.368	19.531	1.714	0.413	94.3924	19.5432
H04	90.055	18.402	4.021	1.077	90.0698	18.4134
H05	90.324	21.274	3.363	0.419	90.2588	21.2476
H06	96.036	28.522	0.946	0.118	95.9941	28.4779
H07	96.225	23.594	0.788	0.086	96.3033	23.697
P01	94.472	18.675	1.606	0.38	94.4737	18.6871
P02	97.117	19.146	1.247	0.474	97.1021	19.1293
P03	98.31	22.377	0.943	0.157	98.3009	22.3564
P04	93.345	18.587	2.543	0.393	93.3218	18.5828
P05	92.129	16.426	2.396	1.871	91.9682	16.3533
P06	97.147	26.625	0.968	0.069	97.1325	26.6045
P07	97.331	19.902	2.024	1.061	97.3008	19.8552
H01'	95.152	19.858	1.874	0.665	95.1356	19.8478
H02'	94.203	22.891	2.26	0.19	94.2081	22.9011
H03'	96.12	22.638	0.927	0.255	96.0043	22.5295
H04'	92.527	19.088	2.06	0.191	92.4458	19.0428
H05'	95.81	23.107	1.248	0.183	95.7229	23.0146
H06'	94.529	21.582	1.233	0.098	94.4605	21.5546
H07'	94.696	21.268	1.498	0.108	94.6711	21.2475
H08'	92.602	19.305	1.772	0.541	92.5769	19.2974
H09'	94.579	22.84	1.297	0.231	94.6333	22.8955
H10'	88.252	18.811	3.51	0.203	88.2338	18.8082
HW1	89.5	15.985	5.219	4.389	89.4569	15.9647
HW2	95.894	24.2	1.03	0.029	95.8759	24.1839
HW3	94.454	20.84	1.402	0.136	94.4407	20.8324
HW4	93.662	19.263	1.598	0.555	93.4817	19.1605
HW5	95.218	19.571	1.889	2.471	95.2245	19.5792
HW6	92.274	19.41	2.297	0.644	92.3577	19.4661
HW7	95.101	23.927	1.195	0.089	95.0364	23.8824
HW8	94.822	23.098	1.182	0.034	94.8611	23.1328
PR1	94.072	17.157	2.997	2.333	94.0032	17.1102
PR2	80.507	12.707	10.61	31.19	80.5598	12.7347
PR3	95.762	18.151	1.44	0.292	95.6525	18.0474
PR4	93.238	18.416	3.005	0.164	93.1658	18.3732
PR5	93.415	17.431	2.015	1.067	93.2731	17.3486
PR6	88.321	19.02	7.055	3.541	88.3361	19.034
PR7	93.832	25.025	2.1	0.252	93.8634	25.0453
PR8	88.349	15.283	3.234	0.455	88.0973	15.213
H01"	95.773	23.094	2.364	0.124	95.6691	22.982
H02"	85.38	16.468	5.572	1.017	85.3794	16.4716
H03"	89.709	18.89	5.621	1.775	89.7757	18.9235
H04"	94.639	22.867	1.632	0.176	94.5684	22.8112
H05"	95.647	22.767	1.188	0.357	95.574	22.6934
H06"	94.602	21.406	2.078	0.923	94.5299	21.3458
H07"	92.791	20.2	1.728	0.202	92.7321	20.1852
H08"	95.807	22.167	1.335	0.244	95.6965	22.0434
H09"	95.593	20.032	1.666	0.341	95.574	20.0161
H10"	94.388	19.623	2.656	0.484	94.3902	19.6288
H11"	93.726	19.769	2.564	0.923	93.7105	19.7629
H12"	95.065	22.404	1.235	0.091	94.9853	22.3325
H13"	92.315	21.333	1.97	0.284	92.3651	21.37
H14"	96.57	24.988	0.895	0.016	96.5542	24.9721
Average	93.583	20.675	2.325	1.187	93.55	20.654

We first compared the performance of our algorithm using the DIBCO'09 documents[3]. These documents were of two types: manuscript (H01 to H07) and printed (P01 to P07). The ground truth was provided by the contest organizer, so the evaluation was considered objective.

In order to assess the sensitivity of the algorithms with respect to their tuning parameters we ran two types of experiments: in the first test we used the same parameters for all the images (figure 11, top). In the second test, these parameters were adjusted for each image (figure 11, bottom). In the first kind of tests (without tuning), the parameters were set according to the recommendations given by the authors (we suggest the readers to see the original papers). For the second kind of tests, the whole range of possible parameter combinations was exhaustively explored. Note that there are two special cases: Otsu's algorithm has no tuning parameter and two of the three parameters of Fabrizio's algorithm can be adjusted automatically. Incidentally, in order not to degrade this method's results, this 'automatic tuning' was not disabled in the first kind of tests.

As we can see, our algorithm, which depends on a unique parameter defined by the value of K (section II-A2 and II-B), clearly outperforms other methods, both with and without parameter tuning. When using the same parameter for all the images, the average of the F-measures for our algorithm is equal to 94.6% (Chen: 75.48, Otsu: 78.74, Ramirez: 83.20, Sauvola: 86.34, Fabrizio: 91.07). When using parameter tuning, this result increases slightly to 95.16% (Chen: 91.14, Sauvola: 91.71, Ramirez: 91.90, Fabrizio: 92.61) which clearly indicates that our algorithm is not too sensitive to the value of K – the weight of each S-FAIR sub-processes).

The results of our method (using the expectation-maximisation algorithm or the K-Means algorithm) are summarized in table II using the new comparison criteria also used in DIBCO11 [22]. We keep the same parameter for each image.

We have also applied the different algorithms on several images with severe degradations. Since there is no ground truth available, the comparison is based on visual considerations. Some results and original image are given in figure 12. Based on visual criteria, the proposed binarization gives the best results on heavily degraded documents. Our solution looks less sensitive to background noise but more sensitive to the text details. Indeed, thanks to Canny's algorithm, background artifacts or variations (due to shadows for instance) are removed right from the start. Furthermore, the clustering process add some robustness around edges. The combination of these two processes gives our algorithm the ability to deal with highly degraded documents. Finally, the last step (labeling the uniform areas as Background or Text) adds to our solution a multi-scale behavior because the size of uniform areas does

[3] We also included the sample images (H06, H07, P06 and P07) available at: http://users.iit.demokritos.gr/~bgat/DIBCO2009/samples/

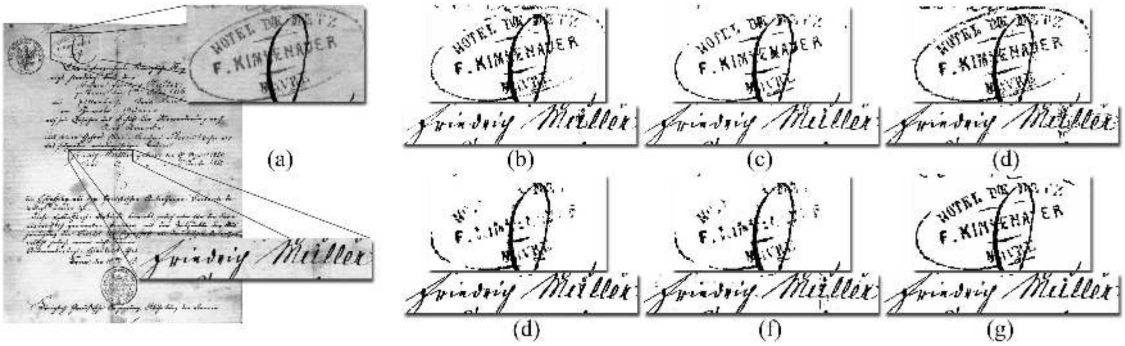

Fig. 12. Example of old manuscripts binarization. (a) Original, (b) Otsu's method [1], (c) Ramirez's method [14], (d) Chen's method [12], (e) Fabrizio's method [16], (f) Sauvola et al. method [4], (g) Proposed method

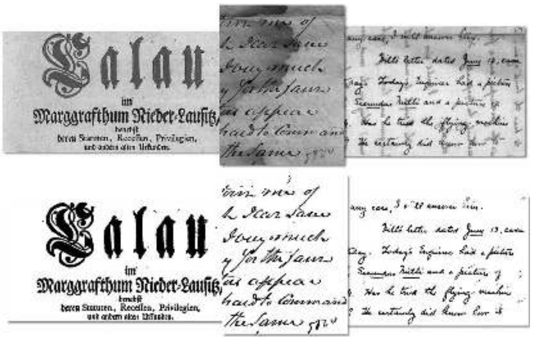

Fig. 13. Some binarization output of the proposed method.

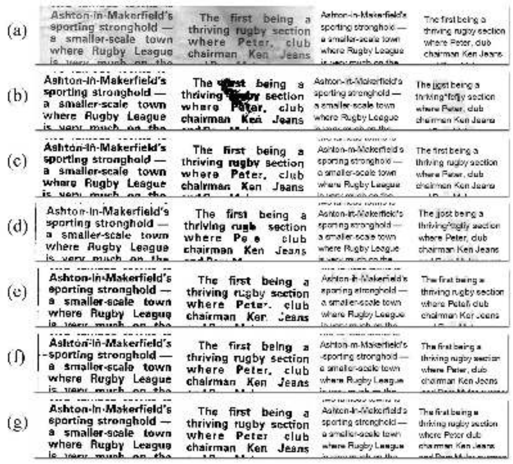

Fig. 14. Document image portion and its corresponding OCR results. (a) Original, (b) Otsu's method [1], (c) Ramirez's method [14], (d) Chen's method [12], (e) Fabrizio's method [16], (f) Sauvola et al. method [4], (g) Proposed method

TABLE III
LEVENSHTEIN DISTANCE FROM THE GROUND TRUTH

Methods	Distance	Recall percent
Proposed method	497	97.487 %
Ramirez	550	96.953 %
Chen	685	96.205 %
Sauvola	700	96.122 %
Fabrizio	744	95.879 %
Otsu	1489	91.752 %

not matter. Our solution is hence very precise around edges while working well with big fonts. Other results are given in figure 13.

In order to compare the recognition rate using an OCR, we use the MediaTeam Oulu Document Database [32] and recent scanning of books and magazines. Some documents were artificially degraded: we down-sampled the gray-scale input images, added some noise, and drew shadows and lights.

We evaluated the performances of the different algorithms by comparing the Levenstein distance from the ground-truth and the results obtained with the well-known OCR engine ABBYY FineReader 9.0 [28]. In figure 14 we give an example of results obtained on a document. With this example, we can see that our method produces a noise-free document while extracting the characters correctly.

The results obtained on the sixteen documents are summarized in table III. These results show the good performances of the algorithms, except for the Otsu method. This bad behavior is due to the global parameter which is not correctly estimated when documents contain pictures or non-uniform illumination. We can also mention that Sauvola's method cannot deal with documents having various font sizes due to the size of the analysis window, so titles are often not correctly recognized.

IV. CONCLUSION

In this paper, we present an efficient algorithm for the binarization of seriously degraded manuscripts. The advantages of this algorithm are multiple:
(1) it is simple in principle and hence easy to implement; (2) the main parameter is directly linked to text detection sensibility and gives good results in most cases; (3) it is efficient for various types of images (manuscript, typewritten, natural scene) and can cope with different contents (font sizes or types, background intensity, ...); (4) it is suitable for parallel implementation since it is SIMD[4] compatible and can be implemented in dedicated massively parallel processor (Xetal IC3D for instance). Indeed, several SIMD implementations of each step have been proposed in the literature (see for instance [33] for the Canny algorithm, [34] for the connected component estimation and [35] for the

[4] Single instruction, multiple data

clustering step).
We have compared this algorithm with several methods in the literature. The evaluation has been carried out on various kinds of manuscripts or typographic documents and is based on four different criteria: the computational time, the F-measure, some visual considerations and the result an OCR algorithm. In conclusion, with respect to the last three quality criteria, our approach outperforms all the other methods. The good performances are coupled with low computation time since this algorithm is roughly ten times as fast as some recent techniques and only twice as slow as the optimized version of the Sauvola algorithm.

For future work, we plan to develop algorithms for parallel architectures and speed up estimation of local means.

References

[1] N.Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[2] J. Bernsen, "Dynamic thresholding of grey-level images," in *Proc. Eighth Int 'l Conf. on Pattern Recognition*, pp. 1251–1255, 1986.

[3] Niblack, *An introduction to digital image processing*. Prentice Hall (July 1986), 1986.

[4] J. Sauvola and M. Pietikinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[5] B. Su, S. Lu, and C. L. Tan, "Binarization of historical document images using the local maximum and minimum," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, (New York, NY, USA), pp. 159–166, ACM, 2010.

[6] B. Gatos, I. Pratikakis, and S. J. Perantonis, "Adaptive degraded document image binarization," *Pattern Recognition*, vol. 39, no. 3, pp. 317–327, 2006.

[7] B. Su, S. Lu, and C. L. Tan, "Combination of document image binarization techniques," in *ICDAR*, pp. 22–26, 2011.

[8] T. Lelore and F. Bouchara, "Document image binarisation using markov field model," in *10th International Conference on Document Analysis and Recognition (ICDAR'09)*, vol. 0, (Los Alamitos, CA, USA), pp. 551–555, IEEE Computer Society, 2009.

[9] J. G. Kuk, N. I. Cho, and K. M. Lee, "Map-mrf approach for binarization of degraded document image," *ICIP08*, pp. 2612–2615, 2008.

[10] N. R. Howe, "Document binarization with automatic parameter tuning," *International Journal on Documant Analysis and Recognition*, 2012. to appear; DOI: 10.1007/s10032-012-0192-x.

[11] R. Cao, C. L. Tan, Q. Wang, and P. Shen, "Segmentation and analysis of double-sided handwritten archival documents," in *Proc. 4th IAPR International Workshop on Document Analysis Systems (Rio de Janeiro)*, pp. 147–158, 2000.

[12] Q. Chen, Q.-s. Sun, P. Ann Heng, and D.-s. Xia, "A double-threshold image binarization method based on edge detector," *Pattern Recognition*, vol. 41, no. 4, pp. 1254–1267, 2008.

[13] Y. Li, C. Y. Suen, and M. Cheriet, "A threshlod selection method based on multiscale and graylevel co-occurrence matrix analysis," in *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*, (Washington, DC, USA), pp. 575–579, IEEE Computer Society, 2005.

[14] M. A. Ramírez-Ortegón, E. Tapia, L. L. Ramírez-Ramírez, R. Rojas, and E. Cuevas, "Transition pixel: A concept for binarization based on edge detection and gray-intensity histograms," *Pattern Recognition*, November 2009.

[15] M. Block and R. Rojas, "Local contrast segmentation to binarize images," in *ICDS '09: Proceedings of the 2009 Third International Conference on Digital Society*, (Washington, DC, USA), pp. 294–299, IEEE Computer Society, 2009.

[16] J. Fabrizio, M. Beatriz, and M. Cord, "Text segmentation in natural scenes using toggle-mapping," in *IEEE International Conference on Image Processing (ICIP09)*, pp. 2373 – 2376, IEEE Computer Society, 2009.

[17] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 679–698, November 1986.

[18] Google, "Google goggle." http://www.google.com/mobile/goggles/#text, 2010.

[19] B. Gatos, K. Ntirogiannis, and I. Pratikakis, "ICDAR 2009 document image binarization contest (DIBCO 2009)," in *ICDAR '09: Proceedings of the 10th International Conference on Document Analysis and Recognition*, (Washington, DC, USA), pp. 1375–1382, IEEE Computer Society, 2009.

[20] R. Paredes, E. Kavallieratou, and R. D. Lins, "ICFHR 2010 contest: Quantitative evaluation of binarization algorithms," in *in Proceedings of International Conference on Frontiers in Handwriting Recognition 2010 (ICFHR10)*, pp. 733 – 736, 2010.

[21] I. Pratikakis, B. Gatos, and K. Ntirogiannis, "H-DIBCO 2010 - handwritten document image binarization competition," in *Proceedings of International Conference on Frontiers in Handwriting Recognition 2010 (ICFHR10)*, pp. 727 – 732, 2010.

[22] B. Gatos, K. Ntirogiannis, and I. Pratikakis, "ICDAR 2011 document image binarization contest (DIBCO 2011)," in *ICDAR '11: Proceedings of the 11th International Conference on Document Analysis and Recognition*, (Washington, DC, USA), IEEE Computer Society, 2011.

[23] I. Pratikakis, B. Gatos, and K. Ntirogiannis, "H-DIBCO 2012 - handwritten document image binarization competition," in *Proceedings of International Conference on Frontiers in Handwriting Recognition 2012 (ICFHR12)*, 2012.

[24] M. Fang, G. Yue, and Q. C. Yu, "The study on an application of otsu method in canny operator," in *International Symposium on Information Processing (ISIP)*, pp. 109–112, Aug 2009.

[25] H. Yuan-Kai, W. Gen, Z. Yu-Dong, and W. Le-Nan, "An adaptive threshold for the canny operator of edge detection," in *Image Analysis and Signal Processing 2010 (IASP10)*, pp. 371–374, 2010.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Royal Statistical Society, Series B*, vol. 39, pp. 1–38, 1977.

[27] L. He, Y. Chao, K. Suzuki, and K. Wu, "Fast connected-component labeling," *Pattern Recogn.*, vol. 42, no. 9, pp. 1977–1987, 2009.

[28] ABBYY, "Finereader 9.0." http://www.abbyy.com/, 2008.

[29] K.-L. Chung and C.-L. Tsai, "Fast incremental algorithm for speeding up the computation of binarization," *Applied Mathematics and Computation*, vol. 212, no. 2, pp. 396–408, 2009.

[30] F. Shafait, D. Keysers, and T. M. Breuel, "Efficient implementation of local adaptive thresholding techniques using integral images," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 6815 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, p. 6, Jan. 2008.

[31] J. Sauvola and H. Kauniskangas, "Mediateam document database ii, a cd-rom collection of document images." http://www.mediateam.oulu.fi/MTDB/, 1999.

[32] G. Bert, D. Francis, and V. Peter, "Implementation of canny edge detection on the wica smartcam architecture," in *3rd ACM/IEEE International conference on Distributed Smart Cameras (ICDSC)*, pp. 1 – 8, 2009.

[33] K. Hawick, A. Leist, and D. Playne, "Parallel graph component labelling with gpus and cuda," *Parallel Computing*, vol. 36, no. 12, pp. 655 – 678, 2010.

[34] B. Hong-tao, H. Li-li, O. Dan-tong, L. Zhan-shan, and L. He, "K-means on commodity gpus with cuda," *Computer Science and Information Engineering, World Congress on*, vol. 3, pp. 651–655, 2009.