UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

STATISTICAL ANALYSIS APPLIED TO DATA CLASSIFICATION AND IMAGE FILTERING

POR
MARCOS ANTONIO MARTINS DE ALMEIDA

Tese submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como parte dos requisitos para a obtenção do grau de Doutor em Engenharia Elétrica

Orientador: Prof. Dr. Rafael Dueire Lins.

RECIFE, Dezembro/2016

Federal University of Pernambuco

Post-graduate program in Electrical Engineering

STATISTICAL ANALYSIS APPLIED TO DATA CLASSIFICATION AND IMAGE FILTERING

for

MARCOS ANTONIO MARTINS DE ALMEIDA

Thesis presented to UFPE as a partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering

Supervisor: Prof. Dr. Rafael Dueire Lins.

RECIFE, December/2016

I dedicate this thesis to my family

1. Now faith is the substance of things hoped for, the evidence of things not seen.

2. For by it the elders obtained a good report.

3. Through faith we understand that the worlds were framed by the word of God, so that things which are seen were not made of things which do appear.

[Hebrews 11]

AGRADECIMENTOS

Agradeço a Deus, fonte de vida, amor e justiça em minha vida.

À minha família, minha mãe Dalva (in memoriam) e minhas irmãs Marilene e Maridalva e cunhados Gesildo e Freitas (in memoriam), pelo carinho e atenção, como também a minha esposa Suely Almeida, pela paciência, dedicação e apoio, juntamente com meus filhos Débora Almeida e Victor Almeida, fontes de motivação na minha vida. Ao amigo e orientador prof. Dr. Rafael Dueire Lins, pela oportunidade de participar do seu grupo de alunos orientandos e poder desfrutar do seu grande conhecimento e de sua vasta cultura e que contribuiu sobremaneira para o meu aprendizado em Processamento de Imagens e áreas afins. O seu exemplo como Professor e orientador me contagiou e motivou para a elaboração dessa tese. Ao professor Fernando Campello pelos valiosos conselhos e orientações ainda na época mestrado. Ao amigo, irmão na fé e professor do Departamento de Eletrônica e Sistemas da UFPE, Frederico Dias Nunes pelo grande incentivo e apoio. Aos professores do Programa de Pós-Graduação em Engenharia Elétrica da UFPE, pela dedicação e atenção dispensados durante o curso, e que muito contribuiram para o meu aprendizado. Aos funcionários do PPGEE, em especial a Andréa Tenório, secretária do Programa, pela atenção e orientação nos assuntos formais e administrativos do curso. À Profa. Sidney Ann Pratt pela verificação e correção dos textos iniciais da tese.

Muito Grato a todos.

Marcos Antonio Martins de Almeida

RESUMO

Análise estatística é uma ferramenta de grande aplicabilidade em diversas áreas do conhecimento científico. Esta tese faz uso de análise estatística em duas aplicações distintas: classificação de dados e processamento de imagens de documentos visando a binarização. No primeiro caso, é aqui feita uma análise de diversos aspectos da consistência da classificação de pesquisadores sêniores do CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, na área de Ciência da Computação. A segunda aplicação de análise estatística aqui desenvolvida trata da filtragem da interferência frente-verso que surge quando um documento é escrito ou impresso em ambos os lados da folha de um papel translúcido. Neste tópico é inicialmente feita uma análise da qualidade dos mais importantes algoritmos de binarização levando em consideração parâmetros tais como a intensidade da interferência frente-verso, a difusão da tinta no papel e a textura e escurecimento do papel pelo envelhecimento. Um novo algoritmo para a binarização eficiente de documentos com interferência frente-verso é aqui apresentado, tendo se mostrado capaz de remover tal ruído em uma grande gama de documentos. Adicionalmente, é aqui proposta a binarização "inteligente" de documentos complexos que envolvem diversos elementos gráficos (figuras, diagramas, etc).

Palavras-chave: Processamento de Dados, Classificação de Dados, Filtragem de Imagens.

ABSTRACT

Statistical analysis is a tool of wide applicability in several areas of scientific knowledge. This thesis makes use of statistical analysis in two different applications: data classification and image processing targeted at document image binarization. In the first case, this thesis presents an analysis of several aspects of the consistency of the classification of the senior researchers in computer science of the Brazilian research council, CNPq - Conselho Nacional de Desenvolvimento CientÃŋfico e TecnolÃşgico. The second application of statistical analysis developed in this thesis addresses filtering-out the back to front interference which appears whenever a document is written or typed on both sides of translucent paper. In this topic, an assessment of the most important algorithms found in the literature is made, taking into account a large quantity of parameters such as the strength of the back to front interference, the diffusion of the ink in the paper, and the texture and hue of the paper due to aging. Also in this topic, a new binarization algorithm is proposed, which is capable of removing the back-to-front noise in several kinds of documents. Additionally, this thesis proposes a new concept of "intelligent" binarization for complex documents, which besides text encompass several graphical elements such as figures, photos, diagrams, etc.

Keywords: Data Processing, Data Classification, Image Filtering.

Contents

List of Figures

List of Tables

1 INTRODUCTION

This Chapter presents the global motivation for the work developed in this thesis that shows applications of statistical analysis to different areas of knowledge. The structure of the document is also outlined.

1.1 Motivations

Data analysis is a process of inspecting, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making. The data can be worked, manipulated, interpreted, and transformed into meaningful conclusions drawn from empirical research studies. In this thesis, two different applications of statistical analysis are addressed with the use of data analysis.

This thesis used advanced hierarchical models that are employed in realistic data analysis. Details of prerequisites and the contents of the thesis are presented as follows. The motivation for the first application was to investigate and verify whether the scientific production of senior researchers in computer science that hold CNPq scholarships is compatible with their classification. In addition, it aims to verify if there is any kind of institutional or regional bias in it. The researchers were re-assessed using data structure analysis and clustering methods such as the *principal component analysis*, the *discriminant analysis*, and the *k-means* having as input data from ArnetMiner (ArnetMiner, 2016), Google Scholar (Scholar, 2016), Microsoft Academic (Academic, 2016), Scopus (Scopus, 2016), the Web of Science (of Science, 2016), and data in the CNPq Lattes (Lattes, 2016) Curriculum Vitae of each of the researchers. The different re-classifications were compared with the one by CNPq and the mismatches and incongruences analysed. Minitab 17 was used as the statistical tool for the tests performed.

The second application of statistical analysis made here is used to assess the quality of the algorithms used for binarizing documents with back to front interference, a phenomenon that appears whenever a document is handwritten or typed on both sides of translucent paper. Such a noise is often found in historical documents, in which the

paper ages and becomes darker, increasing the degree of difficulty in the noise filtering process. One of the important contributions of this thesis is to show that no binarization algorithm is able to perform such a filtering efficiently to all kinds of documents as the degree of the noise intensity, the hue of the paper background and the opacity of the paper vary drastically from document to document.

The experience gained with the assessment of the algorithms to binarize documents with back-to-front interference provided the motivation to attempt to develop a new algorithm for such a purpose using a *bilateral filter*, an image filtering technique that smoothens the image while preserving the edges. Statistical techniques such as linear prediction were applied in the automatic calculation of the threshold value of the filter.

The widespread use of document and image editing tools today has drastically changed the complexity of documents, which often encompass not only text but also all sorts of graphical elements such as photos, histograms, pie (or pizza) diagrams, etc. No global binarization algorithm can be capable of being a *all-case-winner*.

This thesis proposes a new scheme for the binarization of complex documents that decomposes the image in blocks, classifies each of them, and introduces the concept of *semantic binarization*. The idea here is to preserve image information. For instance, the binarization of a pie diagram would cause a complete loss of the original information as the color information is lost. Colors are replaced by different textures to preserve the original information.

1.2 Methodology

The two applications worked on in this thesis have one thing in common: both applications can be represented by a vector of characteristics that qualifies the object to be studied. The first application, the characteristic vector represents the level of scientific production of the researcher, formed by a set of parameters defined by each database. The second application, the characteristic vector represents the texture of the historical document. The methodology used in this thesis in both applications were treated with multivariate statistical analysis tools.

1.3 The Structure of This Thesis

This Thesis has six chapters including this introduction, which presents the motivation for the work developed.

Chapter 2 makes use of cluster analysis techniques to analyze the consistency of the classification of senior researchers in Computer Science of CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), the Brazilian Research Council. The use of multivariate statistical techniques applied to data from international public databases were compared with the CNPq classification. General statistics were presented as: number of researchers by region, by institution, by level of classification, by scientific production, etc.

Chapter 3 assesses the quality of images generated by several binarization algorithms found in the technical literature. The images analyzed in this study belong to the bequest of documents of Joaquim Nabuco held by the Joaquim Nabuco Foundation, a social science research centre in Recife, Brazil, and images from the SBrT (Sociedade Brasileira de Telecomunicações) dataset developed in the LiveMemory Project, led by this thesis supervisor.

Chapter 4 presents a new algorithm for filtering out the back-to-front interference in documents. The new filtering proposed uses a global technique performed in four steps: Filtering using the Bilateral filter, split image in RGB components, decision-making block for each RGB channel based an adaptive binarization method inspired by Otsu's method with a minute choice of thresholding level and classify the binarized images that analyze and to decide which of the RGB components best preserved the text information in the foreground.

The quality of the binary images was assessed *quantitative* and *qualitatively* both in chapters 3 and 4 of this thesis. The quantitative analysis was made by calculating the co-occurence probability matrices of the pixels. Visual inspection was used in the qualitative analysis. Both assessment methods have led to similar conclusions in terms of the efficiency of filtering techniques.

Chapter 5 presents the analysis of the complex documents encompassing several graphic elements such as photos, histograms, pie (or pizza) diagrams besides text. The scheme

automatically decomposes a document image and identifies each of the graphical elements to provide a suitable binarization for each of them independently. The main principle of the scheme proposed here was the recognition that each of the blocks in the complex document has a different nature and no binarization algorithm that does not take such information into account has the slightest chance of succeeding in keeping the fundamental information of the original document. Most binarization algorithms are suitable for scanned text documents and do not work adequately with complex documents that encompass various graphic elements simultaneously.

This thesis presents a new binarization algorithm that works on complex documents. Each of the elements in the image is processed depending on its nature, thus their binarization takes that into account to preserve the original content. The goal is to investigate algorithms that can classify such graphic elements and to find a new "intelligent" dithering algorithm that takes into account the content of the photo, logos, pie diagrams to be binarized.

Chapter 6 draws the conclusions a presents lines for further work along the lines of this thesis.

2 COMPARATIVE ANALYSIS OF A RESEARCH ASSESSMENT

This Chapter analyzes the consistency of the assessment for senior researchers in Computer Science of CNPq. Information from the public databases ArnetMiner (ArnetMiner, 2013), Web of Science (of Science, 2013), Scopus (Scopus, 2013), Lattes (Lattes, 2013), Google Scholar (Scholar, 2013) and Microsoft Academic (Academic, 2013) were compared with the CNPq classification. Several statistical classification strategies were considered. Regional and institutional information were also taken into account.

2.1 Introduction

Assessing research is a challenging task. A number of measures have been developed aiming at assessing the scientific production of researchers and their impact. The number of citations a given paper or author has is an evidence of its/his importance. The h-index, introduced by Hirsch (2005), is defined as the number of papers with citation number higher or equal to h, while the rest of the N papers have less than h citations each. The g-index was introduced by Leo Egghe in 2006 to refine the h-index. According to Egghe (2006, 2007), the g-index gives more weight to the highly-cited papers. Given a number of papers ranked in a decreasing order according to the citations received, the g-index is the largest number such that the top g articles received (altogether) at least g^2 (g square) citations. A comparative discussion between the g and h indices may be found in reference Costas (2008). Many people have raised questions about the validity and consistency of research assessments and such indices. A paper or an author may have high citation indices for having been proved wrong or for being controversial. The *sociability* of the researcher and of the prestige of his institution of affiliation are some of the factors that influence the indices of a researcher. Labbé (2010) shows in his article that there may be ways of manipulating the h-index and articles, showing the Scigen that is an automatic generator of amazing articles using the jargon of computer science. Scigen is based on hand-written context-free grammar and has been developed by the PDOS research group at MIT CSAIL.

Although error-prone, research assessments need to be made. All funding agencies struggle to make *qualitative* unbiased evaluations, but the difficulty of such process always drops down to a *quantitative* one. Scientific journals have undergone profound changes around the world in recent years in order to increase agility in the publishing process - still considered very slow - and to promote open access, which has not yet advanced at the desired speed, among other objectives, according Glenn Hampson, Hampson (2017), executive director of the National Science Communication Institute and the Open Scholarship Initiative program in the United States. According to the researcher, magazine publishing is part of the large, complex ecosystem of scientific communication, which is poorly defined and has evolved through a variety of disconnected efforts and initiatives. Very easily one finds people who wrote their names in the history of science and that produced only very few works of fundamental importance. Unfortunately, that is not the usual case, and even more so since the last decades of the "twentieth" century in which sciences advance in very small steps pulled by hundreds of researchers in each very specialized areas of knowledge.

The whole point in checking the consistency of a research assessment is to try to verify the degree of fairness of the whole process and to check if there is some sort of intentional or unintentional bias in its results. Assessing research is even more difficult in developing countries such as Brazil, in which the Federal annual budget for funding research in all areas is much smaller than the qualified demand. As a result, there is a fierce competition for research funds. CNPq, actively participates in the formulation, implementation, monitoring, evaluation and dissemination of the National Policy on Science and Technology in Brazil. Among such activities, CNPq provides personal grants to the researchers who excel in their research activities. Although the value paid by such a grant ranges today between US$ 250.00 to US$ 300.00 per month, that grant is seen as a recognition of the quality of the research work of the beholder, being thus a symbol of status and prestige. Researchers holding a CNPq grant also qualify for special research funding. Postgraduate programs and undergraduate courses are also assessed based on the number of staff holding CNPq-research grants. The CNPq research grants today are classified in levels $1A$, $1B$, $1C$, $1D$, and 2, in decreasing order. When the data collection was performed the

CNPq research grants were classified in levels Junior research grants (levels $2A$, $2B$, and $2C$) last 2 or 3 years, while the senior grants ($1A$, $1B$ and $1C$) have a duration of 3 or 4 years. There are two grant award meetings of the board per year. The senior group is the target of this work.

As already mentioned, a qualitative research evaluation is an unfruitful task. The history of mankind has shown that several times the geniuses in art and science were far ahead of their time and were only recognized much later on, sometimes posthumously. Such exceptional situation is completely out of the scope of any attempt to measure scientific production. This work focuses on checking if the scientific production of the senior researchers in computer science holding CNPq grants is compatible with their classification. Besides that, it aims to verify if there is some sort of institutional or regional bias in it. It is important to mention that the evaluation process is made by a board (*Comitê Assessor*) of 6 to 8 senior grant holders that follow an indication of the whole community of senior grant holders. The scientific director of CNPq invites those indicated researchers to join the assessment committee for 1, 2, or 4 years attempting to bring in some regional and institutional representativity. No member of such a board may be reconducted immediately after serving on it, guaranteeing a rotativity of the researchers on the board.

This work uses distinct statistical methods to cluster the data found in different research assessment public databases such as ArnetMiner (ArnetMiner, 2016), Web of Science (of Science, 2016), Scopus (Scopus, 2016), Google Scholar (Scholar, 2016), and Microsoft Academic (Academic, 2016). The universe of senior researchers studied (levels 1A, 1B, and 1C of CNPq) encompasses 78 researchers. Besides those public databases, this assessment also took into consideration the CV-Lattes of those researchers, which is a public *curriculum vitae* used by CNPq and all Brazilian research and postgraduate programs. The data in the CV-Lattes is informed by the researcher, reporting different activities (education, affiliations, articles published in journals and conferences, grants, teaching activities, etc.) following standard fields. The researcher states the veracity of the information declared in his CV-Lattes and may suffer legal penalties in case of false declarations. The CV-Lattes platform automatically checks the DOI of publications (if

available), cross-checks information of supervision activities between the CVs of the advisor and student, demands confirmation of institutions that provide grants, etc. The result of the different clustering strategies using the different databases is compared with the classification of the CNPq grants to verify the degree of agreement of the two methods.

The data from ArnetMiner were collected in the period between 11/18/2013 and 11/19/2013; from the Web of Science, Scopus, and Lattes between 11/20/2013 and 11/27/13; from Google Scholar on 12/02/2013; and from Microsoft Academic in the period between 12/03/2013 and 12/06/2013. Not all the 78 senior researchers were found in all databases. The number of CNPq senior researchers found in each database is as follows: ArnetMiner (75 researchers), Web of Science (78 researchers), Scopus (78 researchers), Lattes (78 researchers), Google Scholar (27 researchers) and Microsoft Academic (58 researchers).

The following data was collected for each of the studied researchers from the different databases for later classification/clustering, if available:

- # citation: The total number of citations to publications by a researcher.
- # publication: - The total number of publications by a researcher.
- h-index: A researcher has index h if h of his N papers have at least h citations each, and the other $(N - h)$ papers have at most h citations each.
- g-index: a variant of the h-index, which takes into account the citation evolution of the most cited papers over time.
- activity: the total number of papers published in the last n years or during the whole carreer.
- diversity: the number of different research fields a researcher has publications in. The author-conference-topic model to obtain the research fields for each expert.
- sociability: The score of sociability is basically defined based on how many coauthors an expert has.

The following indices are present and were used for classification in each of the databases:

- ArnetMiner (AM): activity, number of citations, *h*-index, papers, *g*-index, sociability and diversity.
- Web of Science (WS): number of articles, # citations, *h*-index and medium.
- Scopus (Sc): total number of articles, citations, mean and *h*-index.
- Lattes: articles in scientific journals, number of papers published in proceedings, books authored, book chapters, number of patents, and number of successful Ph.D. supervisions.
- Google Scholar (GS): number of citations, *h*-index, *i10*-index, citations(2008), *h*-index(2008) and *i10*-index(2008).
- Microsoft Academic (MA): number of publications, number of citations, cited by and number of co-authors.

It is also important to observe that besides not all source databases have different coverage of the universe studied of 78 senior researchers, their data also vary as, for instance, the number of papers or citations by a given researcher may vary from database to database, and from them to the Lattes CV, which is informed by the researcher. An analysis of the possibility of existing a bias in grant distribution between the regions is also provided here.

Brazil is geopolitically divided into five regions (also called macroregions) by the Brazilian Institute of Statistics and Geography (IBGE); each region is composed of three or more states. Officially recognized, the division in such regions is not merely an academic and geopolitical one; social and economic factors, including funding are also region-based. The five Regions of Brazil with and their states are:

1. North (N): Acre, Amapá, Amazonas, Pará, Rondonia, Roraima, Tocantins.
2. Northeast (NE): Alagoas, Bahia, Ceará, Maranhão, Paraíba, Pernambuco, Piauí, Rio Grande do Norte, Sergipe.
3. MidWest (CO): Goiás, Mato Grosso, Mato Grosso do Sul, Distrito Federal (Federal District).

4. Southeast (SE): Espírito Santo, Minas Gerais, Rio de Janeiro, São Paulo.

5. South (S): Paraná, Rio Grande do Sul, Santa Catarina.

The Southeast and South regions are the most densely populated and economically developed in Brazil.

To the best of the knowledge of the author of this thesis, no work reports on the analysis of consistency of the assessment of researchers with CNPq grants. The only effort that attempts to do so is the paper by Barata (2003) which analyzes the profile of the researchers in public health with CNPq personal research grants. The analysis considered the researcher's undergraduate and graduate degrees, field of expertise, scientific output, and publications. The validity of the analysis presented here goes far beyond the analysis of the fairness of the distribution of personal research grants in Brazil. It also shows how difficult it is to make a fair and unbiased comparative research assessment.

2.2 Material and Methods: Statistical Tools

A problem that arises quite often in many research areas is that, given a set of n individuals, grouping them into k homogeneous classes or subsets, heterogeneous with each other (i.e., individuals of different subgroups are dissimilar).

There are two kinds of classification procedures: supervised and unsupervised classification. The supervised classification is the essential tool used for extracting quantitative information from object data. Using this quantitative information, the analyst has enough data available to generate representative parameters for each class of interest. This step is called training. Once trained, the classifier is then used to attach labels to all the object data according to the trained parameters. According to Richards (1993), the Maximum Likehood Classification is the most widely used kind of supervised classification used. Its effectiveness depends on how reasonably accurate is the estimation of the mean vector m and the covariance matrix for each spectral class data.

On the other hand, unsupervised classification does not require *a priori*, knowledge of the classes, making use of some clustering algorithm to classify the data. These procedures can be used to determine the number and location of the unimodal spectral classes. One

of the most commonly used unsupervised classification method is the *migrating means clustering*. Richards (1993) describes this method as initially assigning each object one of the n-clusters which have as central objects n-randomly chosen ones. Then, the distances of all objects to the centers of all the clusters is calculated. Objects are moved to minimize the sum of the square error. Such a procedure is iteratively repeated until stability is reached.

In attempting to choose an appropriate analytical technique, to classify the CNPq Senior researchers, three methods were used: the principal component analysis approach, (Anderson, 1984), the Discriminant Analysis, (Hair, 2009), that is used to determine which variables are the best predictors and the k-means method, (Hair, 2009).

2.2.1 The Principal Component Analysis Method

The Principal component analysis is appropriate when one has obtained measures on a number of observed variables and wishes to develop a smaller number of artificial variables (called *principal components*) that will account for most of the variance in the observed variables. The principal components may then be used as a predictor or criterion variables in the subsequent analyses.

A Principal component analysis focuses in explaining the variance-covariance structure of a set of variables through a few linear combinations of such variables. Its general objectives are: data reduction and interpretation. A Principal component can be defined as a linear combination of optimally-weighted observed variables.

The following procedure will perform a Principal Components Analysis on a set of data.

- Step 1: Get some data set; $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_n$.
- Step 2: Subtract the mean; The mean subtracted is the average across each dimension. This produces a data set whose mean is zero.
- Step 3: Calculate the covariance matrix; The formula for covariance is always mea-

sured between 2 dimensions. The formula for covariance is:

$$cov(X,Y) = \frac{\sum_1^n (X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)} \tag{2.2.1}$$

- Step 4: Calculate the eigenvectors and eigenvalues of the covariance matrix; Recall that covariance is always measured between 2 dimensions. If one has a data set with more than 2 dimensions, there is more than one covariance measurement that can be calculated.

  The covariance matrix for an imaginary 2 dimensional data set, using the usual dimensions x, y, has 2 rows and 2 columns, and the values are these:

$$\begin{bmatrix} cov(x,x) & cov(x,y) \\ cov(y,x) & cov(y,y) \end{bmatrix} \tag{2.2.2}$$

- Step 5: Choosing components and forming a feature vector.

In general, once eigenvector are found from the covariance matrix the next step is to order them by eigenvalue, the highest to the lowest. This gives the components in order of significance. Now, one can decide to ignore the components of lesser significance.

If one leaves out some components, the final data set will have less dimensions than the original. More precisely, if one originally has n dimensions in the data, one calculates n eigenvectors and eigenvalues, and then one chooses only the first p eigenvectors, then the final data set has only p dimensions.

Algebraically, the principal components are linear combinations of the p random variables $X_1, X_2, \ldots, X_p$. Geometrically, these linear combinations represent the selection of a new coordinate system obtained by rotating the original system with $X_1, X_2, \ldots, X_p$ as the coordinate axes, according to Hair (2009).

2.2.2 The Discriminant Analysis Method

Discriminant analysis is a method that can be applied for data classification and selection. The methods of discriminant analysis are based on the assumption that a

subdivision of the data is available, and the aim is to seek directions in space showing the separation of such subgroups or to determine a rule for future classifications. But according to Crammer (2003), often there is no classification of its type available, and the problem is to identify which (and how many) are the different classes of existing individuals in the data set available.

In the following, the linear discriminant analysis method is discussed, in which the classification problem can be solved by finding the linear functions that best divide the groups into clusters.

The aim of Discriminant Analysis is to develop an equation that will help to predict the value of a dependent variable based on the values of a set of independent variables. The dependent variable is qualitative. The dependent variable must be nonmetric, representing groups of researches that are expected to differ on the independent variables. The dependent variables are chosen as the most representative variables of the groups of interest.

The choice of a dependent variable are the best representing of groups interest.

Discriminant analysis is a parametric technique to determine which weights for the quantitative variables or predictors best discriminate between two or more groups of cases and do so better than chance, according to Crammer (2003). The analysis creates a discriminant function which is a linear combination of the weights and scores on such variables. The maximum number of functions is either the number of predictors or the number of groups minus one, whichever of these two values is the smallest.

$$Z_{jk} = a + w_1 X_{1k} + w_2 X_{2k} + \cdots + w_n X_{nk}, \tag{2.2.3}$$

where:

$Z_{j,k}$ = discriminant Z score of discriminant function j for object k. a = intercept. w_i = discriminant weight for independent variable i. X_{ik} = independent variable i for object k.

Whenever there are several groups, the discriminant functions offer a separation threshold, but it is unable to predict the number of members in each of the classes. Thus, in addition to improving the explanation of group membership, these additional discrimi-

nant functions add insight into the various combinations of independent variables that discriminate between groups. With three categories of the dependent variable, discriminant analysis can estimate two discriminant functions, each representing a different dimension of discrimination. Thus, one can now calculate two discriminant scores for each respondent.

Discriminant analysis can be used to classify observations into two or more groups if one has a sample with known groups. Discriminant analysis can also be used to investigate how variables contribute to group separation.

2.2.3 The k-Mean Method

The aim of the k-means algorithm is to divide m points in n dimensions into k clusters such that the sum of the squares is minimized within clusters.

The k-mean is the most popular partitioning method of clustering. It was firstly proposed by MacQueen(1967) (abud Johnson (2007)). It is an unsupervised, non-deterministic, numerical, iterative method of clustering. In k-mean each cluster is represented by the mean value of objects in the cluster. Here, a set of n objects is divided into k clusters in such a way that the intercluster similarity is low and intracluster similarity is high. Similarity is measured in terms of mean value of objects in a cluster, according to Yadav (2013).

The k-mean procedure assigns each item to the cluster having the nearest centroid (mean). In its simplest version, the process is composed of the following steps:

1. Partition the items into k initial clusters.

2. Proceed through the list of items, assigning an item to the cluster whose centroid is the nearest (the distance is usually computed using the Euclidean distance). Recalculate the centroid for the cluster that receives the new item and for the cluster loses the item.

3. Repeat step 2 until no more reassignments take place.

The k-means clustering algorithm works best when sufficient information is available to make good starting cluster assignments. The final assignment of the items to a cluster

will be, to some extent, dependent upon the initial partition or the selection of the points, which are taken as *seeds*.

2.3 Results

The following results were obtained from the databases using the statistical methods of analysis outlined.

2.3.1 General Statistics

The representativeness of the databases analysed is a fundamental starting point. The volume of data (published articles of all researchers analysed) and the number of citations to those articles. The databases with the largest volume of cumulative citations is ArnetMiner, as shown in Figure 2.1, the one with the smallest volume is the Web of Science database. The largest volume of cumulative publications is found in the Lattes

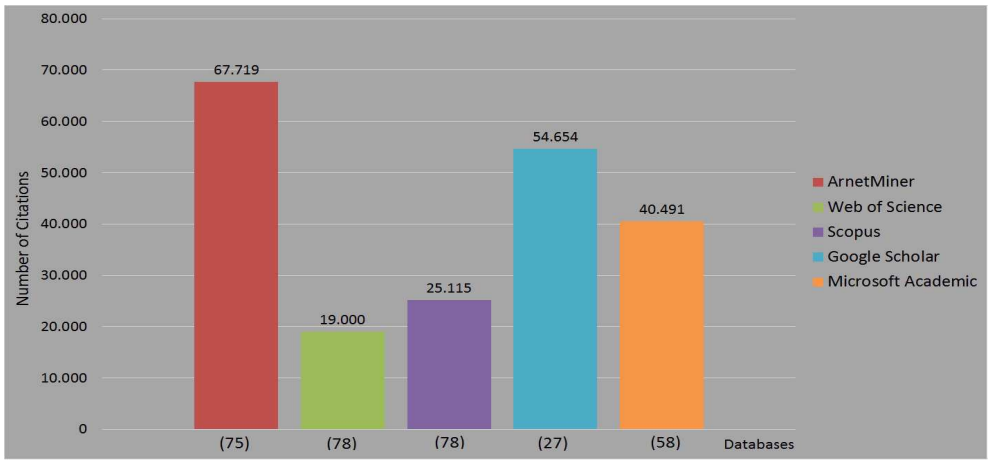


Figure 2.1: Number of citations in each database.

database, which is informed by the researcher himself. The second largest is in the ArnetMiner database. The Web of Science is the smallest in terms of the volume of accumulated citations, as shown in Figure 2.2.

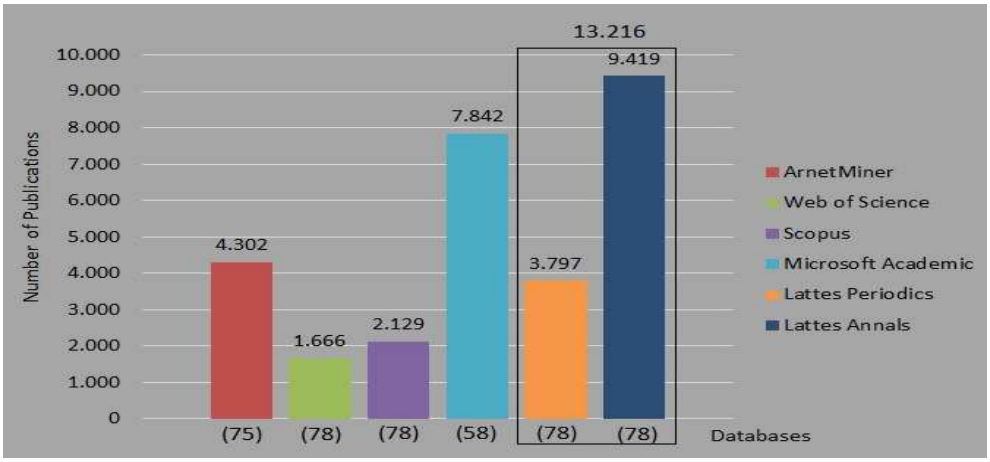


Figure 2.2: Number of publications in each database for the universe of researchers under study.

Figure 2.3 shows the distribution of the Senior researchers in Brazil by the rank of CNPq and by region, in which one may observe that there is a strong concentration in the Southeast, the most economically developed region in Brazil. The Northeast region is represented by the Federal University of Pernambuco (UFPE), while the South region by Federal University of Rio Grande do Sul (UFRGS). The distribution of researchers

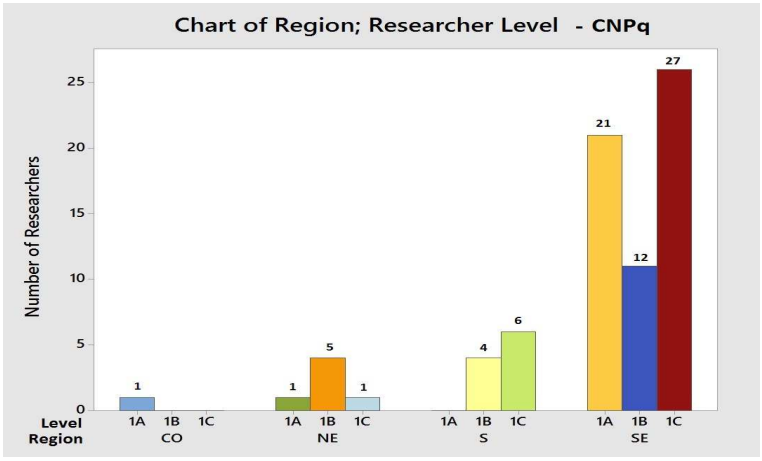


Figure 2.3: Distribution of Senior researchers by CNPq level and regions of Brazil. Source: Lattes database.

by institution is shown in Figure 2.4. The largest number of top researchers ($1A$) at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio) with eight researchers, followed by the University of São Paulo (USP) and the Federal University of Rio de Janeiro (UFRJ), with seven researchers $1A$-level each.

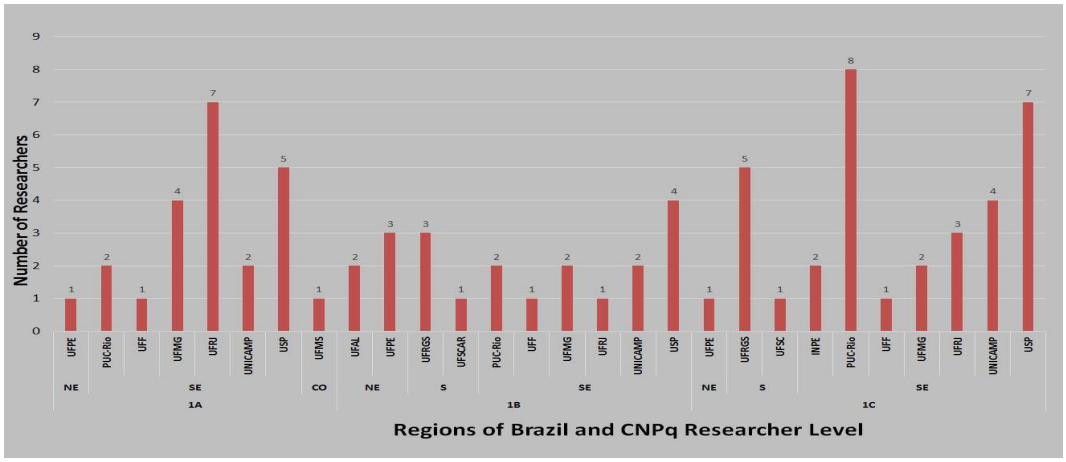


Figure 2.4: Number of researchers by institution in Brazilian regions. Source: Lattes database.

Figure 2.5 shows the scientific production of researchers studied by the classification levels of CNPq, the total articles published in journals, the number of doctoral supervisions completed, as well as master dissertations directed. Surprisingly under such aspects only,

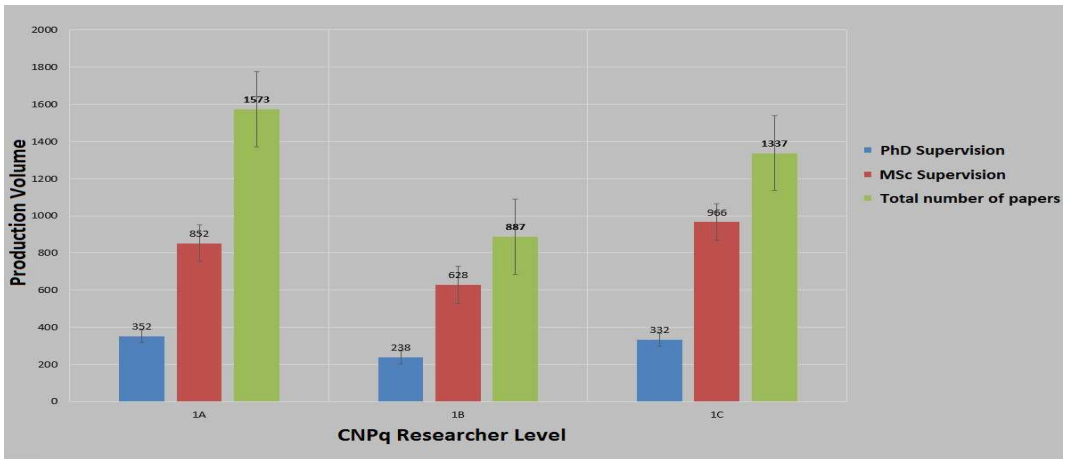


Figure 2.5: Number of doctoral and master supervisions and the total number of articles in periodicals as function of CNPq research level. Source: Lattes database.

the production volume of the researchers classified by CNPq as $1C$ level is higher than that for those classified at the $1B$-level. This may be a sign of some level of inconsistency in the researcher classification.

The per capita production of researchers in the category $1C$ is equivalent to the production of researchers at level $1B$, as shown in Figure 2.6. The time since the completion of the Ph.D. degree in the group of researchers $1C$ is also close to in the group of re-

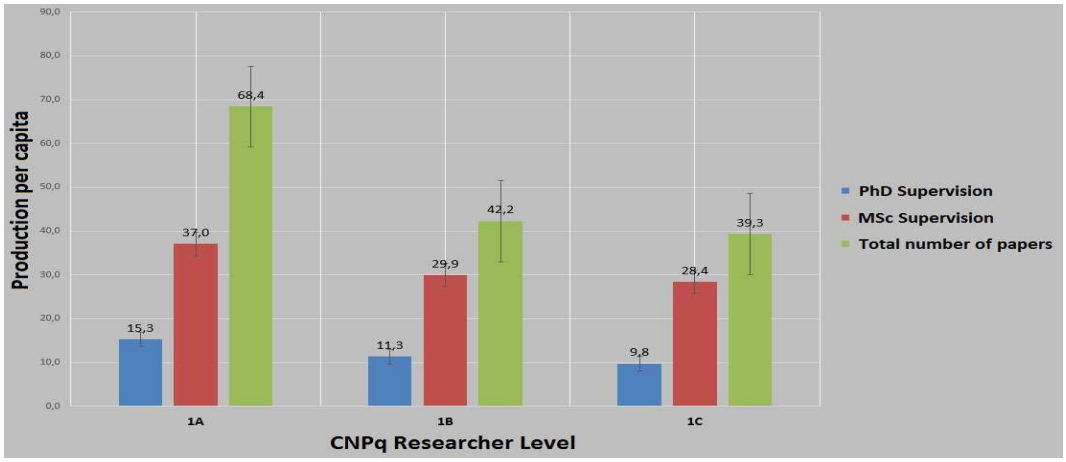


Figure 2.6: Per capita number of completed supervisions of Ph.D. and M.Sc. and the number of articles published in periodicals as function the research of CNPq. Source: Lattes database.

searchers $1B$, as shown in Figure 2.7. Figure 2.8 shows the h-index of the researchers

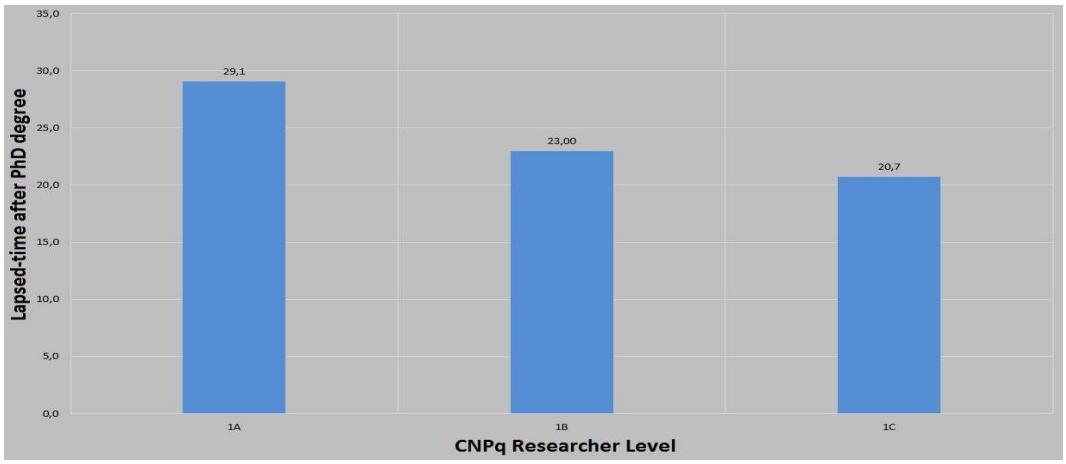


Figure 2.7: CNPq Researcher level versus lapsed-time after PhD degree. Source: Lattes database.

studied. The databases were considered: ArnetMiner, Scopus and Web of Science. One way observe that the h-index of researchers who are at the level $1C$ and $1B$ as equivalent to the Web of Science database. A more detailed statistical analysis follows:

The number of articles published by the researchers studied in the databases shows that the mean values differ from database to database. Therefore, the scales of the plots are not the same. In the following, several databases will be considered. A normality test applied to the databases indicates that the data do not follow a normal distribution, p-value > 0.05. Mood Median test was used to analyze and provided evidence that the

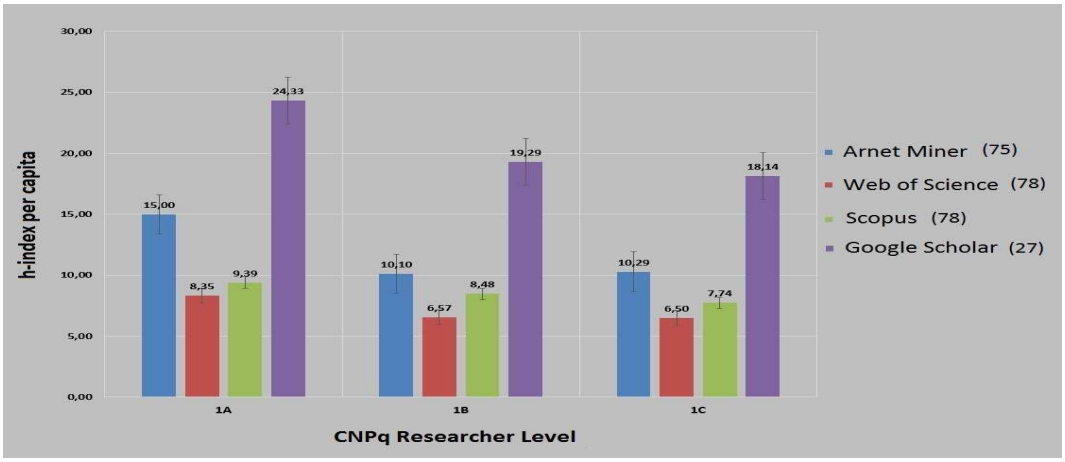


Figure 2.8: h-index per capita. Source: ArnetMiner (AM), Web of Science (WS), Scopus (Sc) and Google Scholar (GS) databases.

means of the groups can be considered equal to the 5% level of significance ($p < 0.05$, rejecting the null hypothesis H_0). The data from the databases Scopus and Microsoft Academic show that the samples come from populations with equal medians. On the other hand, the data from the ArnetMiner and Web of Science databases, come from populations with medians that are not equal, according Figure 2.9.

Figure 2.9 shows that the number of published articles by researcher in all the studied databases have a mean and variance within the same range of values, in absolute terms, except for the outliers.

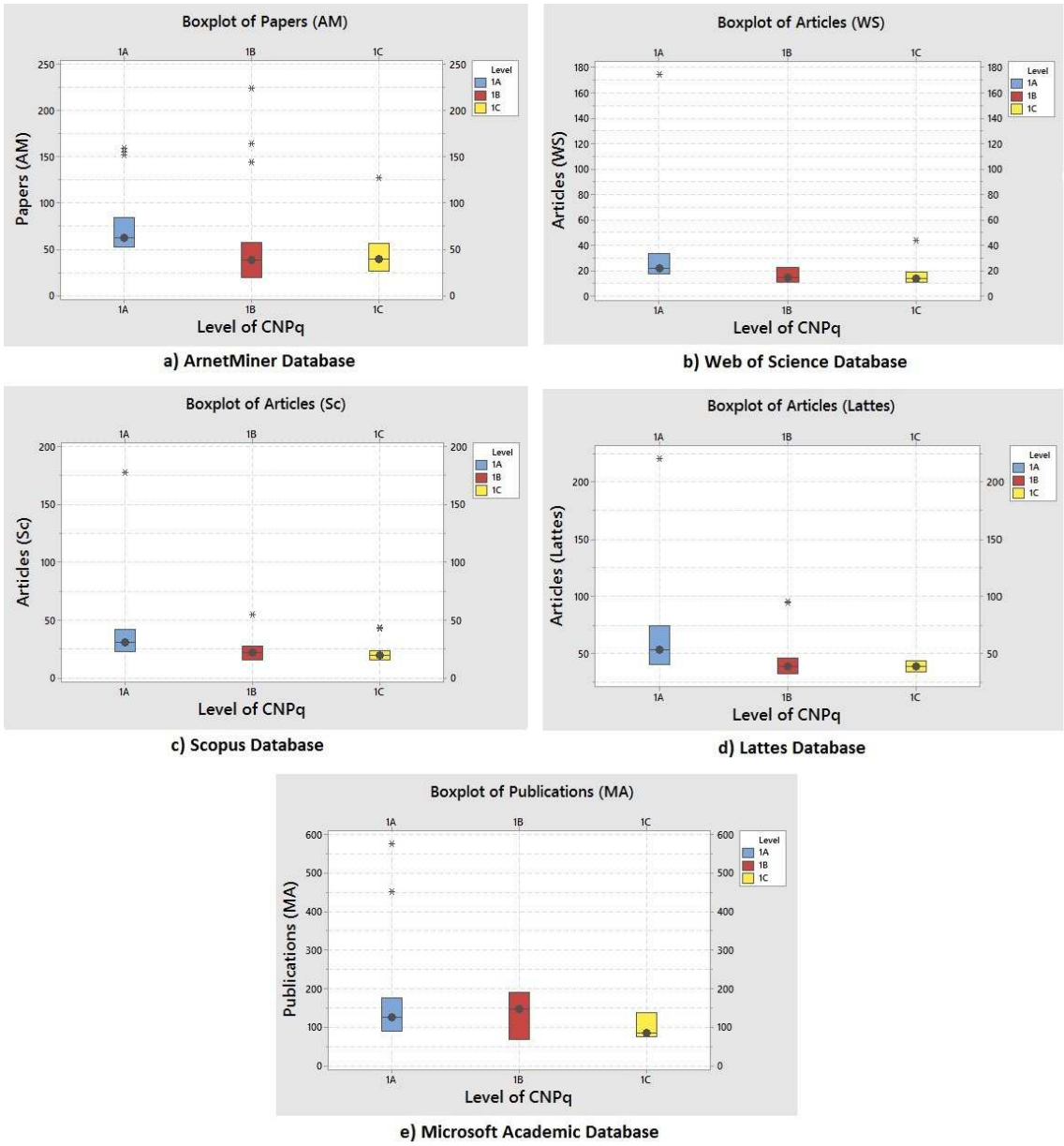


Figure 2.9: Number of articles of different databases.

With respect to the number of citations of articles, the data from the Web of Science, Scopus, Microsoft Academic and Google Scholar databases have samples that come from populations with close means. While the data from the ArnetMiner database come from populations with means that are different, as shown in Figure 2.10.

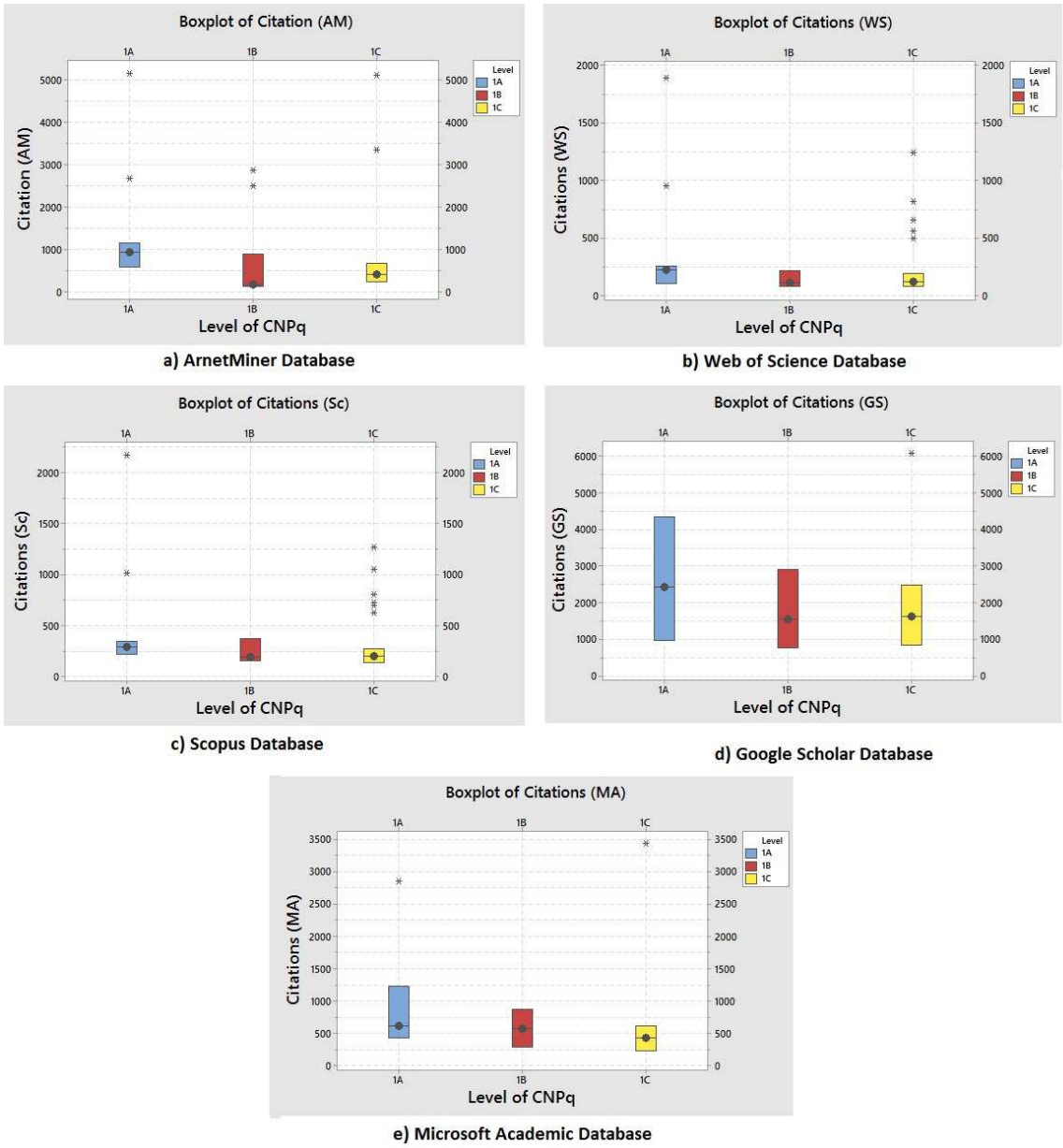


Figure 2.10: The number of citations in the different databases.

There are researchers who have a high production of articles and citations, which appear as outliers in all databases. This is the main reason why they appear in clusters with few researchers when the discriminant analysis and k-mean models are applied.

According to ArnetMiner and Microsof Academic databases, the number of articles authored by the researchers from the South region of the Brazil is higher than in the production of articles from the Southeast; the average number of articles by the researchers of the Northeast is similar to the one of the researchers of the Southeast, as shown in Figure 2.11.

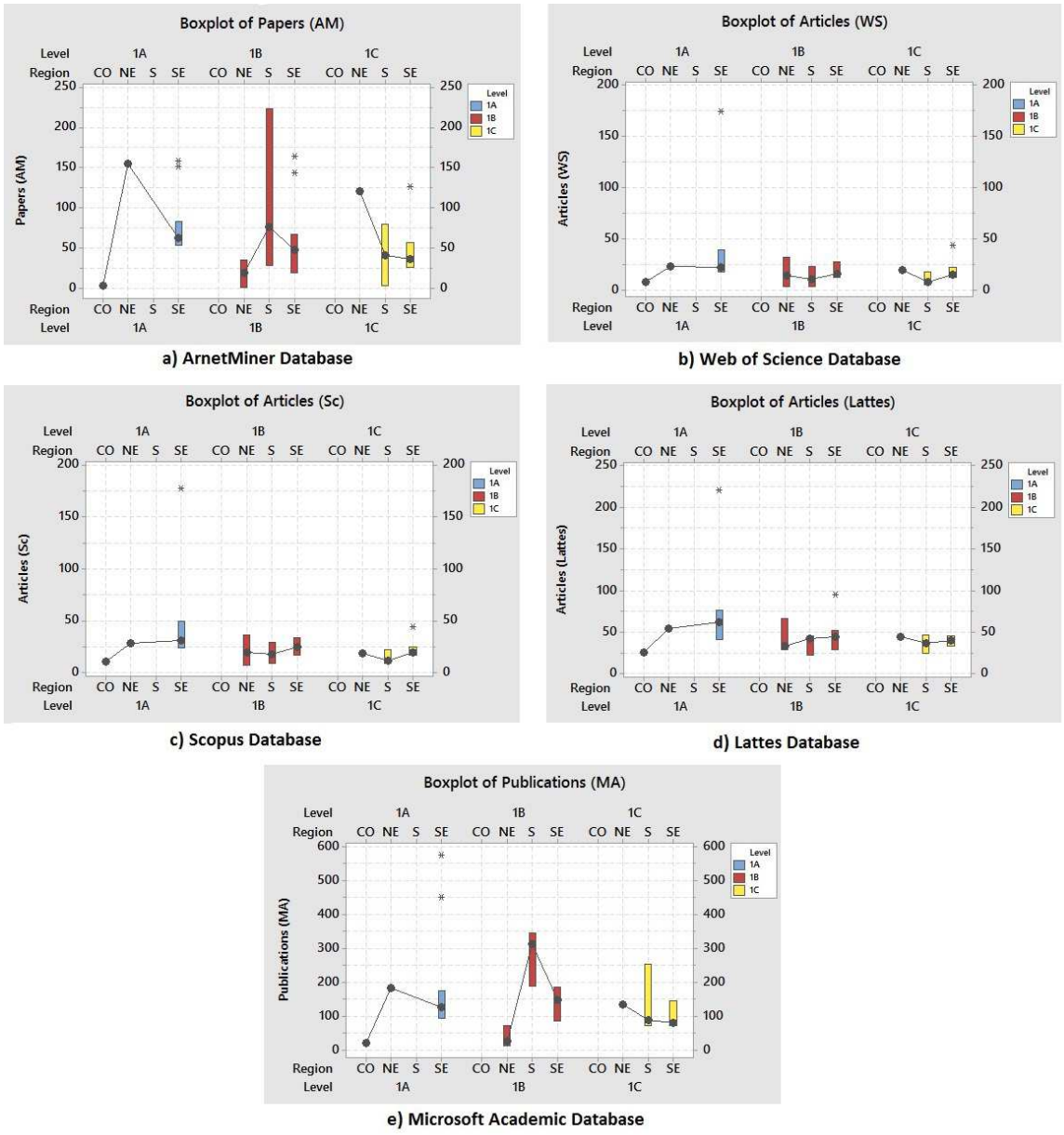


Figure 2.11: Number of articles by the CNPq researchers by region.

Figure 2.12 shows k-index in the different databases.

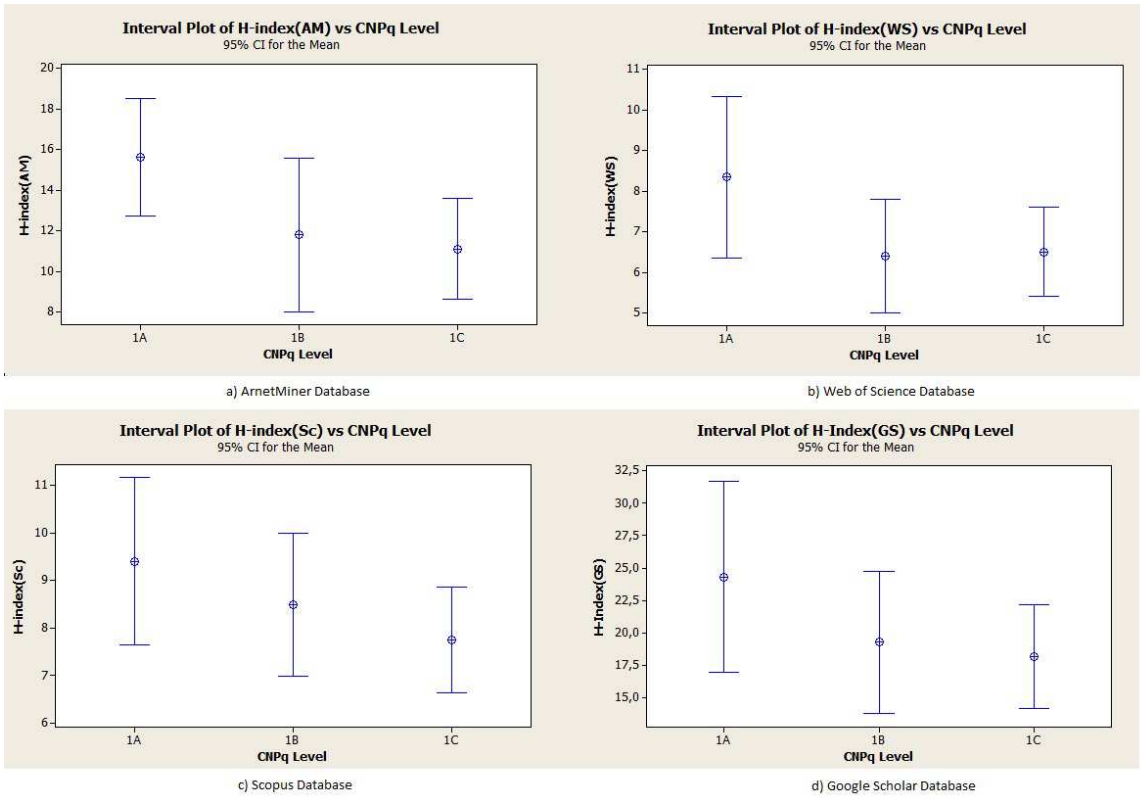


Figure 2.12: The k-index in the different databases.

2.3.2 The Principal Component Analysis Method

Using the Principal Component Analysis (PCA) method one can calculate the principal components of all the studied databases, as shown in Table 2.1. The PCA order depends on the number of variables in each database. The goal of PCA is also to order the clusters by regrouping, and to explain the maximum amount of variance with the fewest number of principal components.

Table 2.1: Eigenanalysis of the correlation matrix of all databases.

Database	Principal Component	1	2	3	4	5	6	7
ArnetMiner								
	Eigenvalue	4.970	0.847	0.506	0.294	0.239	0.114	0.028
	Proportion	0.710	0.121	0.072	0.042	0.034	0.016	0.004
	Cumulative	0.710	0.831	0.903	0.945	0.980	0.996	1.000
Web of Science								
	Eigenvalue	2.929	0.896	0.142	0.031			
	Proportion	0.732	0.224	0.036	0.008			
	Cumulative	0.732	0.956	0.992	1.000			
Scopus								
	Eigenvalue	2.954	0.870	0.148	0.026			
	Proportion	0.739	0.218	0.037	0.007			
	Cumulative	0.739	0.956	0.993	1.000			
Lattes								
	Eigenvalue	2.715	1.188	0.891	0.656	0.317	0.231	
	Proportion	0.453	0.198	0.149	0.109	0.053	0.039	
	Cumulative	0.453	0.651	0.799	0.909	0.961	1.000	
Goolgle Scholar								
	Eigenvalue	5.482	0.245	0.160	0.066	0.031	0.014	
	Proportion	0.914	0.041	0.027	0.011	0.005	0.002	
	Cumulative	0.914	0.955	0.981	0.992	0.998	1.000	
Microsoft Academic								
	Eigenvalue	2.960	0.779	0.198	0.061			
	Proportion	0.740	0.195	0.050	0.015			
	Cumulative	0.740	0.935	0.985	1.000			

The generation of the principal component as a linear combination of the main variables:

$$\begin{aligned} PC1(AM) &= 0.365 * (activity) + 0.382 * (citation) + 0.421 * (h - index) + \\ &+0.384 * (papers) + 0.413 * (g - index) + 0.293 * (sociability) + \\ &+0.374 * (diversity). \end{aligned} \tag{2.3.1}$$

$$\begin{aligned} PC1(WS) &= 0.497 * (articles) + 0.575 * (citation) + 0.343 * (md) + 0.384 * (papers) + \\ &+0.552 * (h - index). \end{aligned} \tag{2.3.2}$$

$$\begin{aligned} PC1(Sc) &= 0.481 * (articles) + 0.570 * (citation) + 0.376 * (md) + 0.384 * (papers) + \\ &+0.551 * (h - index). \end{aligned} \tag{2.3.3}$$

$$\begin{aligned} PC1(Lattes) &= 0.389 * (articles) + 0.434 * (papers) + 0.362 * (books) + \\ &+0.047 * (patents) + 0.517 * (book\ chapters) \\ &+0.510 * (completed\ doctorate\ guidelines). \end{aligned} \tag{2.3.4}$$

$$\begin{aligned} PC1(GS) &= 0.396 * (activity) + 0.417 * (citations) + 0.406 * (h - index) + \\ &+0.416 * (i10 - index) + 0.401 * (citations(2008)) + \\ &+0.413 * (i10 - index(2008)). \end{aligned} \tag{2.3.5}$$

$$\begin{aligned} PC1(MA) &= 0.516 * (publications) + 0.471 * (citation) + 0.461 * (co - authors) + \\ &+0.547 * (cited\ by). \end{aligned} \tag{2.3.6}$$

For each database, there is a variable which has a higher weight. For the Web of Science, Scopus, Google Scholar and Microsoft Academic databases the parameter with the highest weight is the number of citations of papers. While for the ArnetMiner database it is the k-index, and for the Lattes database is the number of book chapters published by the researcher.

Most of the coefficients in the Equations 7.1.39 to 7.1.29 have the same order of magnitude, ranging between 0.4 to 0.5, which means that the variables selected are representative. Otherwise, in the Lattes database the Equation 7.1.42, one can ignore the "number of patents" component, because it not significant, as its weight is 0.047.

The pattern of the data from the Web of Science and Scopus databases is similar, as is the number of citations of articles. The number of citations of the researchers from the Northeast and South regions is similar in the ArnetMiner, Google Scholar and Microsoft Academic databases, as shown Figure 2.13

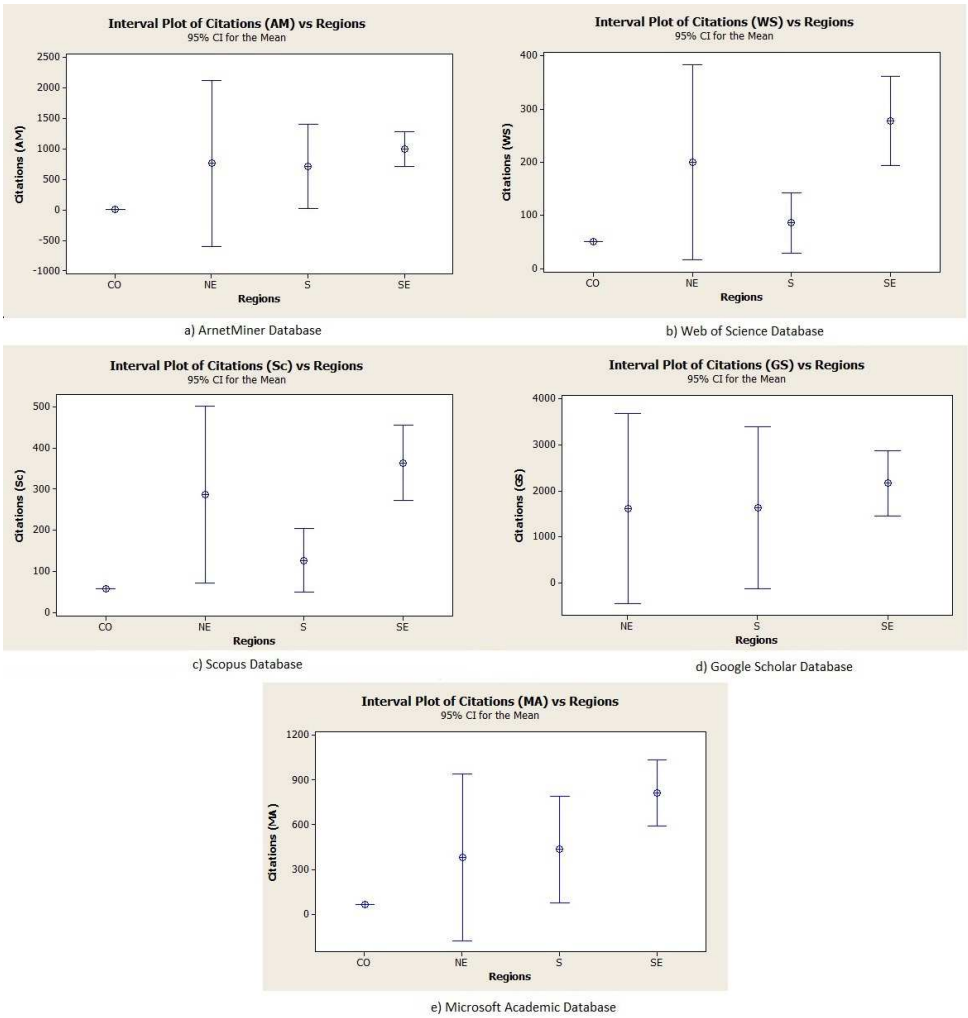


Figure 2.13: Number of citations by researchers from CNPq by region.

Figure 2.14 shows that *h*-index of the researchers from the Northeast region is comparable to the one researchers from the Southeast in the Web of Science, Scopus and Google Scholar databases.

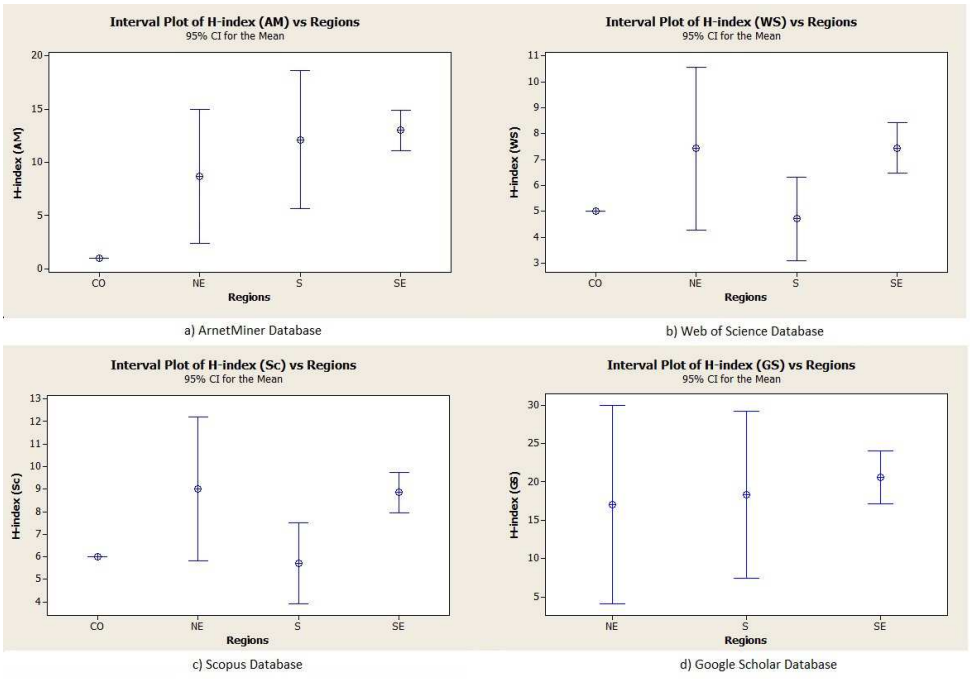


Figure 2.14: h-index of the researchers by Region.

2.3.3 The Discriminant Analysis Method

To minimize the effect of the differences in scale in all the variables, their values were standardized, by subtracting the mean and dividing by the standard deviation before calculating the distance matrix. The cluster centroids and distance measures are in the standardized variable space before the distance matrix is calculated.

Using the linear discriminant analysis, all groups are assumed to have the same covariance matrix. The quadratic discrimination does not make such assumption, however.

The ratio is calculated by the following quotient, used in Tables 2.2, 2.3 and 2.4:

$$Ratio = \frac{N\ Regrouping}{N\ Total} \tag{2.3.7}$$

When the Ratio is equal to 1, it means that the accuracy rate of the method used in the classification of researchers is 100% according to the CNPq rank, for each level. The hit percentage is similar for all databases, observing the quadratic discriminant analysis model shown in Table 2.2. The linear model was run and presented indices with a slightly better proportion.

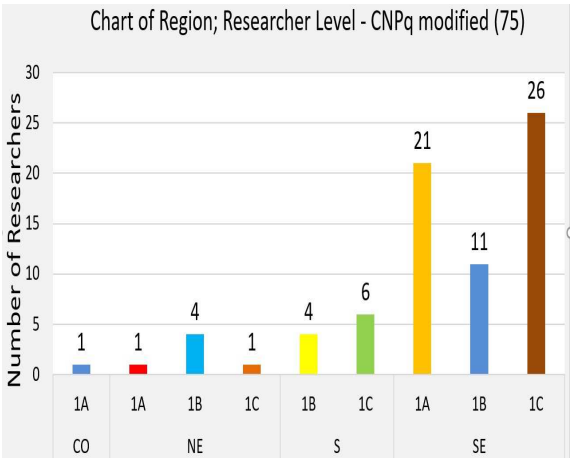
Table 2.2: Comparison of databases using the discriminant analysis method.

Database	1A	1B	1C	Sum	Correct
ArnetMiner					
N Total	23	19	33	75	
N Regrouping	16	07	52	75	
Ratio	0.695	0.368	1.575		
N Correct	12	05	28	45	
Proportion	0.522	0.263	0.848		0.600
Web of Science					
N Total	23	21	34	78	
N Regrouping	11	59	08	78	
Ratio	0.478	2.809	0.235		
N Correct	08	19	07	34	
Proportion	0.348	0.905	0.206		0.436
Scopus					
N Total	23	21	34	78	
N Regrouping	10	45	23	78	
Ratio	0.434	2.142	0.676		
N Correct	09	15	13	37	
Proportion	0.391	0.714	0.382		0.474

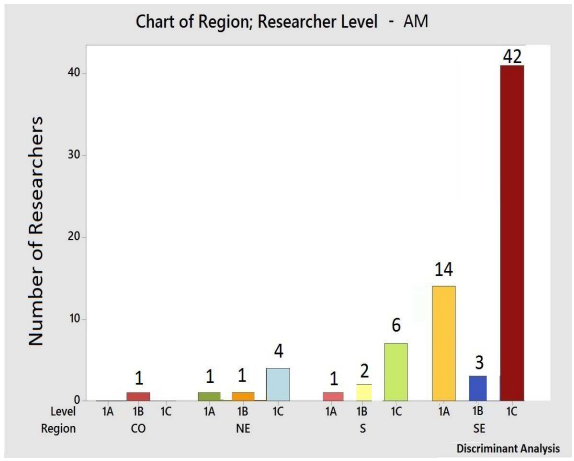
Database	1A	1B	1C	Sum	Correct
Lattes					
N Total	23	21	34	78	
N Regrouping	12	26	40	78	
Ratio	0.521	1.238	1.176		
N Correct	10	12	25	47	
Proportion	0.435	0.571	0.735		0.603
Google Scholar					
N Total	06	07	14	27	
N Regrouping	07	08	12	27	
Ratio	1.166	1.142	0.857		
N Correct	06	06	11	23	
Proportion	1.000	0.857	0.786		0.852
Microsoft Academic					
N Total	15	16	26	57	
N Regrouping	08	14	35	57	
Ratio	0.533	0.875	1.346		
N Correct	05	06	21	32	
Proportion	0.333	0.375	0.808		0.561

The application of discriminant analysis method resulted in a higher success rate in the Google Scholar database, 85.2%. Next, come the Lattes and ArnetMiner databases with 60.0%. The worst indices were obtained with Scopus and Web of Science databases which are around 47.40% and 43.60%, respectively.

Figure 2.15b shows the new distribution of reclassification of CNPq researchers using the discriminant analysis method for the ArnetMiner database. Figure 2.15a shows the distribution of CNPq modified for 75 researchers, the same order of magnitude of the ArnetMiner database to facilitate the comparison between databases.



(a) Distribution of Reclassification of CNPq Modified.

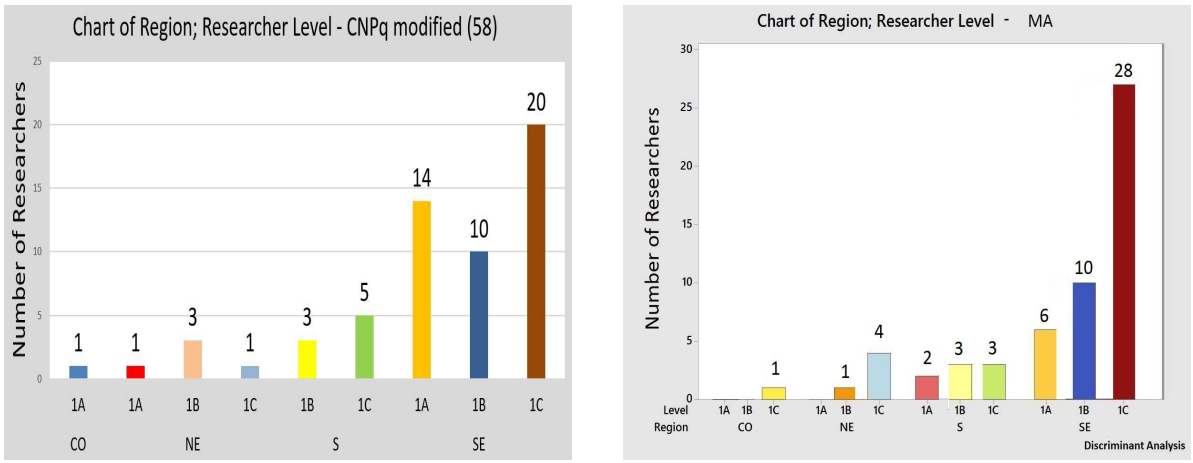


(b) New Distribution of CNPq Researchers.

Figure 2.15: Discriminant analysis approach. Source: ArnetMiner database.

Applying the discriminant analysis with the Microsoft Academic database, was obtained the new distribution of reclassification of researchers from CNPq, shown in Figure

2.16b. The comparison between databases can be made with the CNPq modified database shown in Figure 2.16a.



(a) Distribution of Reclassification of CNPq Modified. *(b)* New Distribution of CNPq Researchers.

Figure 2.16: Discriminant analysis approach. Source: Microsoft Academic database.

2.3.4 The k-Mean Method

The k-mean approach reclassifies researchers from the data arranged in each database, considering the original classification of CNPq as the initial partition. Equations 7.1.39 to 7.1.29, using the k-mean method generated a rank with eight clusters as shown in Table 2.3, enabling the calculation of each cluster and then further regrouping into three clusters.

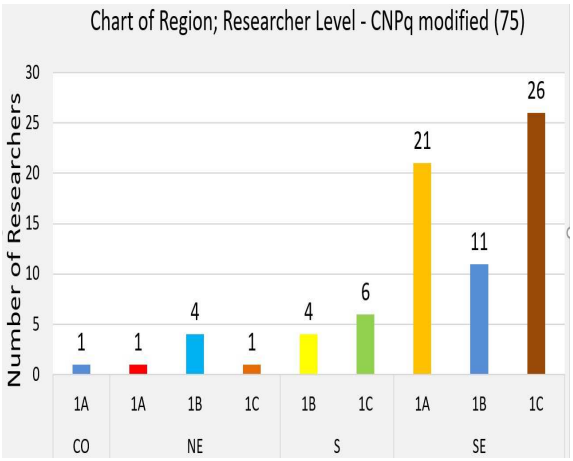
The same method was used for the analysis at the data shown in Table 2.4, except for the regrouping that was performed using four groups. At regrouping, by using the k-means approach, the database that showed the best proportion was Google Scholar. The worst proportion was provided by the Web of Science database, as shown in Table 2.3.

As a result, the new distribution of reclassification of the CNPq researchers, into three groups, using the k-mean approach, was applied to the ArnetMiner database, as shown in Figure 2.17b. The comparison between databases can be made with the CNPq modified database shown in Figure 2.17a.

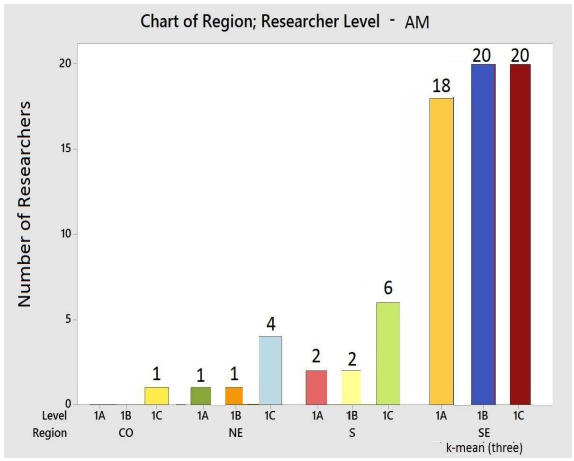
Table 2.3: Regrouping in three groups using k-mean method.

Database	1A	1B	1C	Sum	Correct
ArnetMiner					
N Total	23	19	33	75	
N Regrouping	21	23	31	75	
Ratio	0.913	1.210	0.939		
N Correct	10	06	11	27	
Proportion	0.434	0.315	0.333		0.360
Web of Science					
N Total	23	21	34	78	
N Regrouping	17	23	48	78	
Ratio	0.739	1.095	1.411		
N Correct	08	04	11	23	
Proportion	0.347	0.190	0.323		0.294
Scopus					
N Total	23	21	34	78	
N Regrouping	23	22	33	78	
Ratio	1.000	1.047	0.970		
N Correct	10	07	09	26	
Proportion	0.434	0.333	0.264		0.333

Database	1A	1B	1C	Sum	Correct
Lattes					
N Total	23	21	34	78	
N Regrouping	12	19	47	78	
Ratio	0.521	0.904	0.411		
N Correct	07	06	14	27	
Proportion	0.304	0.285	0.411		0.346
Google Scholar					
N Total	06	07	14	27	
N Regrouping	05	05	17	27	
Ratio	0.833	0.714	1.214		
N Correct	02	02	02	06	
Proportion	0.333	0.285	0.142		0.222
Microsoft Academic					
N Total	15	16	26	57	
N Regrouping	10	11	36	57	
Ratio	0.666	0.687	1.384		
N Correct	03	03	12	18	
Proportion	0.200	0.187	0.461		0.315



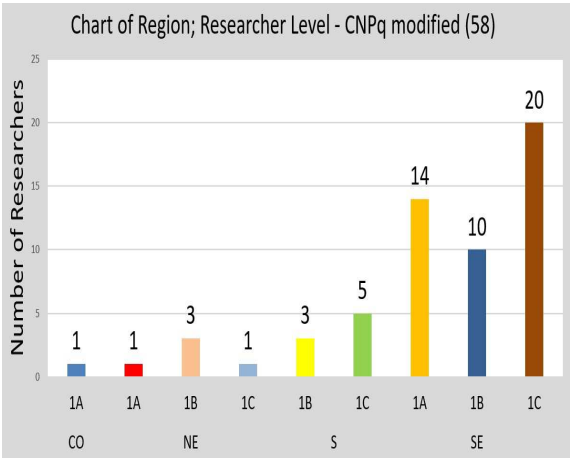
(a) Distribution of Reclassification of CNPq Modified.



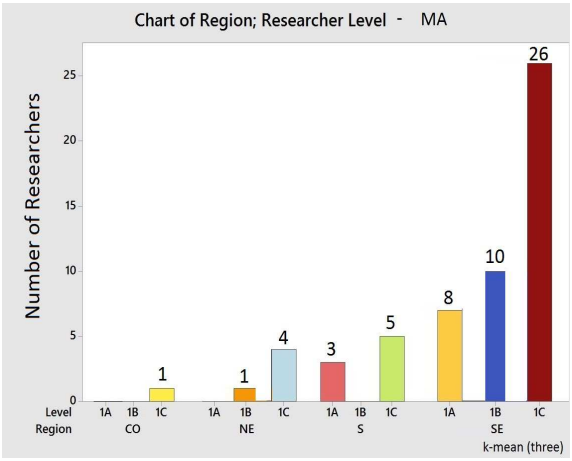
(b) New Distribution of CNPq Researchers.

Figure 2.17: Discriminant analysis approach. Source: ArnetMiner database.

Figure 2.18b shows the reclassification distribution of CNPq researchers, into three groups, using the discriminant analysis method for the Microsoft Academic database.



(a) Distribution of Reclassification of CNPq Modified.



(b) New Distribution of CNPq Researchers.

Figure 2.18: Discriminant analysis approach. Source: Microsoft Academic database.

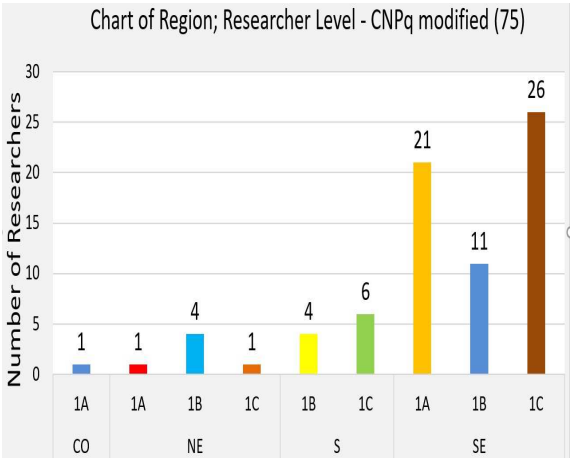
By creating a new grouping, called X, for the researchers who do not meet the criteria of research levels $1A$, $1B$ and $1C$, set at their centroids and by using the k-mean approach, we can make a reclassification into four groups, as shown in Table 2.4: The application of

Table 2.4: Regrouping in four groups using k-mean method.

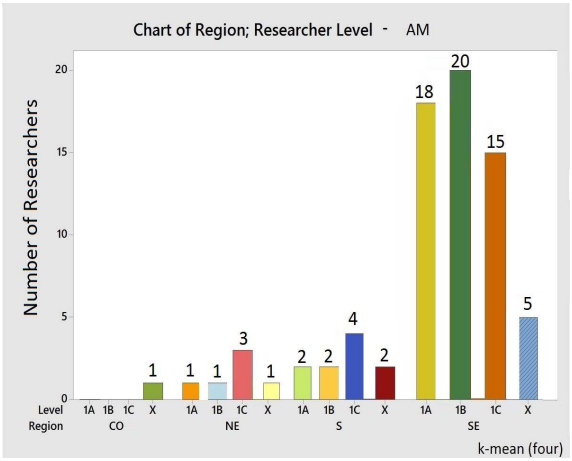
Database	1A	1B	1C	X	Sum	Correct
ArnetMiner						
N Total	23	19	33		75	
N Regrouping	21	23	22	09	75	
Ratio	0.913	1.210	0.667			
N Correct	10	06	11		27	
Proportion	0.434	0.315	0.333			0.360
Web of Science						
N Total	23	21	34		78	
N Regrouping	17	23	26	12	78	
Ratio	0.739	1.095	0.764			
N Correct	08	04	11		23	
Proportion	0.347	0.190	0.323			0.294
Scopus						
N Total	23	21	34		78	
N Regrouping	23	22	19	14	78	
Proportion	1.000	1.047	0.558			
N Correct	10	07	09		26	
Proportion	0.434	0.333	0.264			0.333

Database	1A	1B	1C	X	Sum	Correct
Lattes						
N Total	23	21	34		78	
N Regrouping	12	19	30	17	78	
Ratio	0.521	0.904	0.882			
N Correct	07	06	14		27	
Proportion	0.304	0.285	0.411			0.346
Google Scholar						
N Total	06	07	14		27	
N Regrouping	05	05	05	12	27	
Ratio	0.833	0.714	0.357			
N Correct	02	02	02		06	
Proportion	0.333	0.285	0.142			0.222
Microsoft Academic						
N Total	15	16	26		57	
N Regrouping	10	11	25	11	57	
Ratio	0.667	0.687	0.961			
N Correct	03	03	12		18	
Proportion	0.200	0.187	0.461			0.315

the k-mean method to the ArnetMiner database yielded a new reclassification into four groups as shown in Figure 2.19b. The comparison between databases can be made with the CNPq modified database shown in Figure 2.19a.



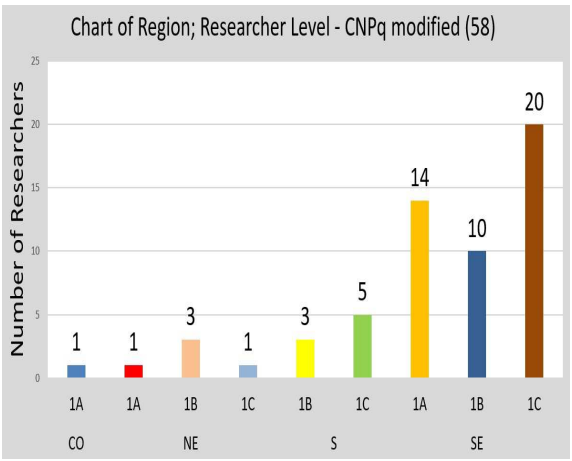
(a) Distribution of Reclassification of CNPq Modified.



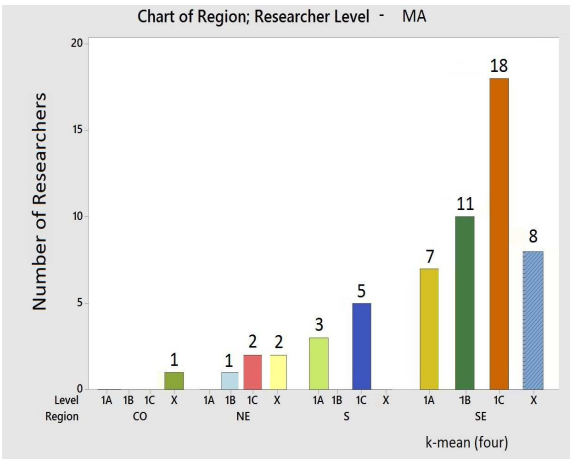
(b) New Distribution of CNPq Researchers.

Figure 2.19: Discriminant analysis approach. Source: ArnetMiner database.

Figure 2.20b shows a new reclassification for researchers of CNPq, when using the k-mean method applied to the Microsoft Academic database.



(a) Distribution of Reclassification of CNPq Modified.



(b) New Distribution of CNPq Researchers.

Figure 2.20: Discriminant analysis approach. Source: Microsoft Academic database.

2.4 Conclusions

When taking into consideration the geographic regions of Brazil, one can see that the distribution of the researchers using the CNPq classification is uneven, with a strong bias towards the Southeast region of Brazil, and with a greater percentage of researchers in relation to the Northeast and South regions, about 70%. The Notheast region represents only 7% of the total number of researchers.

Notably for the Northeast region with regard to the volume of papers, publications by researchers in categories $1A$ and $1C$, are higher than in other regions of the country, taking as reference the ArnetMiner database. The researchers of the South region, belonging to the $1B$ level, had a number of publications above the other researchers from all regions, according to the analysis of the data in the Microsoft Academic database. In the Web of Science, Scopus and Lattes databases, the data values for publications of researchers, independently of the CNPq level, which are close to the average values, except those for the Midwest region.

Using the discriminant analysis approach, and making a comparison between the various databases, we found that the data from the Google Scholar database has the highest proportion of correct hits as 85.2%, for the researchers in classifications $1A$, $1B$ and $1C$, although the number of samples is small. The analysis of the Lattes and ArnetMiner databases provide equivalent results of about 60.0%. The worst hit rate is obtained with the Web of Science database, 43.6%. The highest proportion of classification similarity is using the data provided by the Google Scholar database in the case of the researchers in the $1A$ group, 100.0%. The worst classification results is obtained with the Web of Science database in group $1C$, 20.6%.

Applying the k-mean method, the databases that showed the highest correct proportion were the ArnetMiner and Lattes, with values of 36.0% and 34.6%, respectively. The results of the regrouping shown in Figure 2.21 is a histogram plot, representing the information from all the databases using the discriminant analysis and k-mean methods. For each database, there is a classification by hits.

One can find that the distribution reclassifications in the discriminant analysis method is more conservative because it meant a limited number of grants for researchers $1A$, a

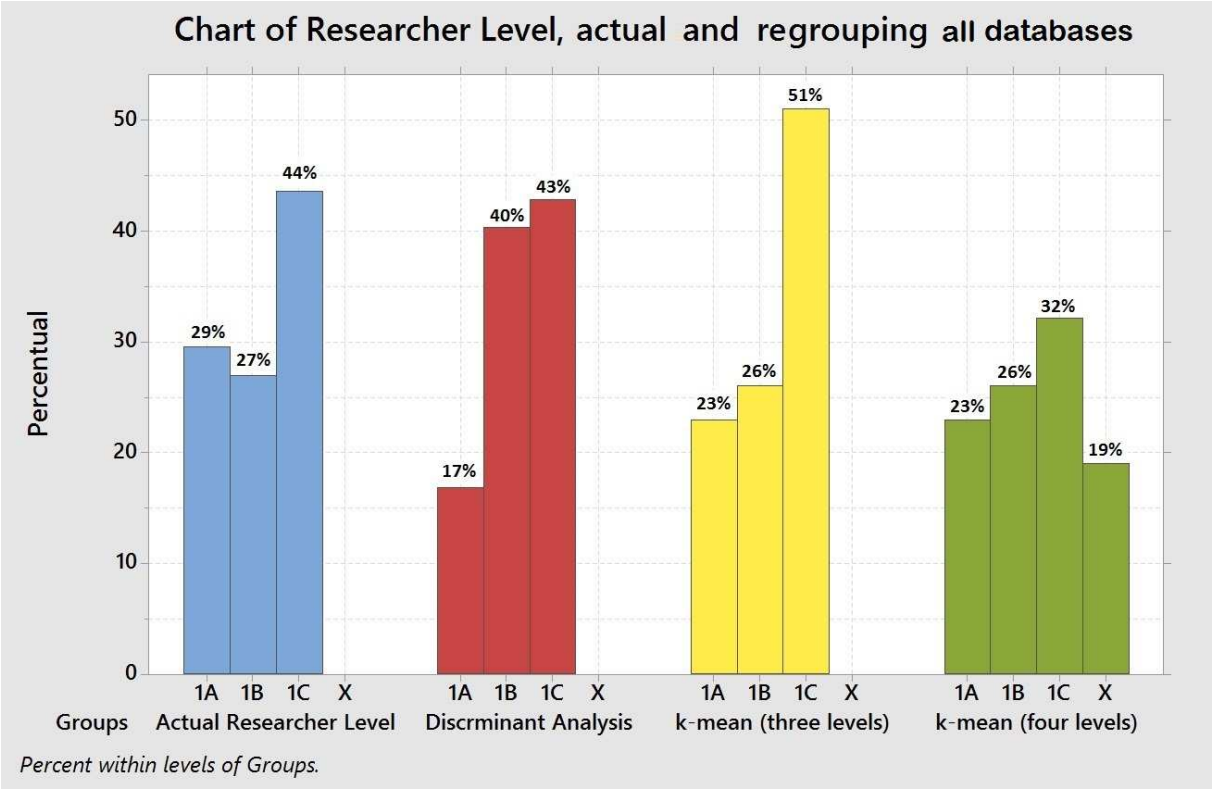


Figure 2.21: Actual CNPq researcher level and regrouping, using discriminant analysis and k-mean methods.

higher number to $1B$ and $1C$. The k-mean method for four levels, has the uniform and restricted distribution.

The discriminant analysis method was effective in identifying the appropriate database for the classification and ordering of researchers for $1A$, $1B$ and $1C$ levels, given the correct percentage of correlation to the CNPq classification. In this case, the ArnetMiner database represents the production of CNPq researchers correctly as well. Considering a set of all databases, the method remained robust with satisfactory results.

The criteria presented by CNPq does not differ much from the results presented by the methods described, discriminant analysis and k-mean. With respect to the ArnetMiner database, it showed a consistency in the set of given variables, considering the volume of data as well as the representative variables able to generate a clustering and reclassification models.

In conclusion one can say that the CNPq rank is fair in general terms as the researcher classification distortions are small.

3 ASSESSING BINARIZATION TECHNIQUES FOR DOCUMENT IMAGES WITH BACK-TO-FRONT INTERFERENCE

Documents written on both sides of translucent paper make visible the ink from one side on the other. This artefact, first described in the literature in reference (Lins, 1995), is called "back-to-front interference", "bleeding", (Kasturi, 2002), or "show-through", (Sharma, 2001). The direct binarization of documents with such interference may yield unreadable documents. The technical literature presents several algorithms for suitably removing such an artefact, but the quality of the response given depends on a number of factors such as the strength of the interference, the hue of the paper with the aging, etc. This chapter assesses several algorithms to remove back-to-front interference, such as: IsoData, Pun, Kapoo-Sahoo-Wong, Johannsen-Bille, Yen-Chang-Chang, Otsu, Mello-Lins, Silva-Lins Rocha and Wu-Lu methods, where the filter input synthesized images representing several historical documents shown in Figures 3.5 and 3.6.

3.1 Introduction

Whenever a document is typed or written on both sides of a sheet of paper and the opacity of the paper is such as to allow the back printing to be visualized on the front side, such as in the Figure 3.1, the degree of difficulty of good segmentation increases enormously. A new set of hues of paper and printing colors appears. If the document is scanned either in true-color or gray-scale, the human eye is able to filter out that sort of noise keeping document readability, as shown in Figure 3.3, which presents a grayscale conversion of the image in Figure 3.1. This is not the case with automatic tools such as OCRs. Binarized images (black and white images) claim less storage space, allow for faster network transmission, and are more suitable to be processed by most commercial OCR tools.

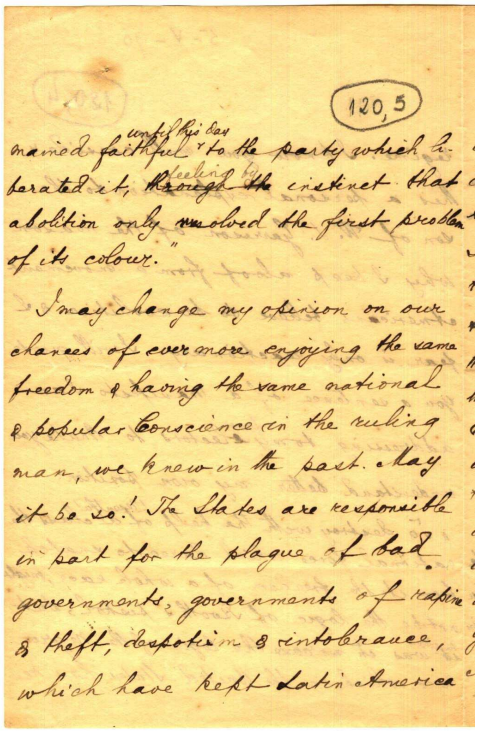
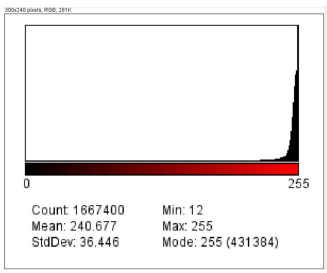
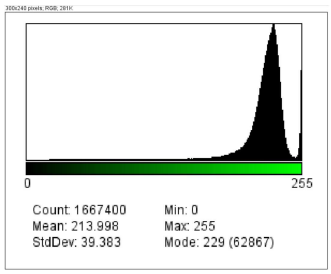


Figure 3.1: Historical Document: A simple example with back-to-front interference.

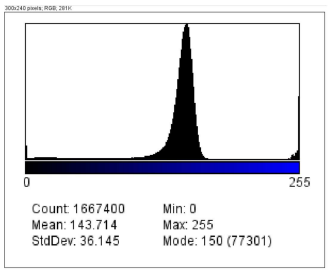
In a document such as the one presented in Figure 3.1, one expects to find three color clusters corresponding to the ink in the foreground, the paper background and the trespassed ink (the back-to-front interference). Unfortunately, no image representation provided such clustering to allow the easy filtering out of the back-to-front interference. The RGB colour histogram may be seen in Figure 3.2 in which one may observe an overlap of the distributions of the background, the ink in the foreground, and the interfering ink from the backside.



(a) Red Colour Histogram of Figure 3.1.



(b) Green Colour Histogram of Figure 3.1.



(c) Blue Colour Histogram of Figure 3.1.

Figure 3.2: RGB Colour Histogram.

The binarized version of this document generated by the direct application of the binarization algorithm by using Jasc Paint Shop Pro ™ version 8 (Palette component: Gray values, Reduction component: nearest color, Palette weight: non-weighted) is completely unreadable, as one may observe in Figure 3.4. As one may also observe in Figure 3.1 the interfering ink of the backside of the paper does not look as sharp as the one in the foreground. There is a kind of blur effect that not only affects the contours of the text but also gives rise to different hues of the ink. Several papers in the literature addressed the back-to-front interference problem. Some authors use waterflow models (Oha, 2005), other researchers have used wavelet filtering (Tan, 2002), but the technique of most widespread use is thresholding (Kavallieratou & Antonopoulou, 2005), (Leedham, 2002) and Wang & Tan (2001). The most successful thresholding techniques for filtering out the back-to-front interference are based on the Shannon's entropy (Abramson, 1963) of the gray-scale document (Mello & Lins, 2000) and (Mello & Lins, 2002). Although recent advances were made in finding efficient algorithms that yield good quality images (Silva, 2006), a final solution in the removal of the back-to-front interference is still sought off.

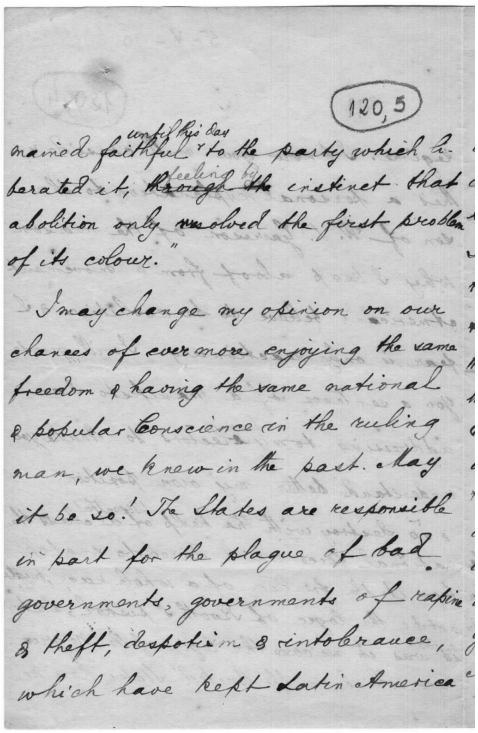


Figure 3.3: Gray-scale version of Figure 3.1.

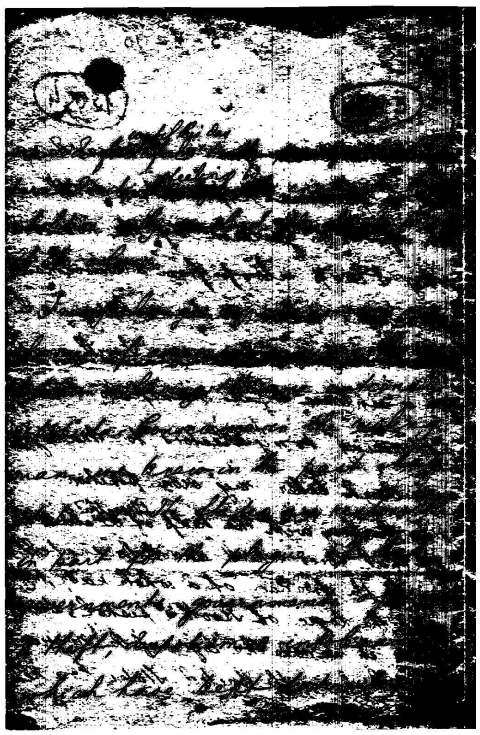


Figure 3.4: Binarized document of Figure 3.1.

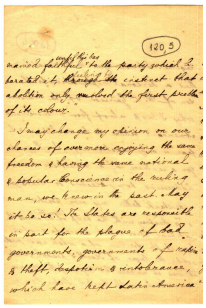
The conversion from true-color image into 256 levels gray-scale images as an intermediate step towards image binarization has shown to be a valuable simplification, adopted by all the binarization algorithms either used or developed for removing back-to-front interference. The first processing step is generating grayscale documents from the true-color ones by using the standard Equation 3.1.1 to calculate the value of the new pixel:

$$Gs = 0.299R + 0.587G + 0.114B, \tag{3.1.1}$$

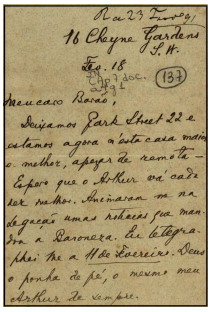
where R, G, and B are the Red, Green and Blue values of the pixel in the true-color image, and the gray-level is the value of the pixel in the grayscale image. As one may observe from the image of the document exhibited in Figure 3.3, the grayscale conversion of documents tends to preserve their readability.

3.2 Generation and Filtering of Images with Synthetic Degradation

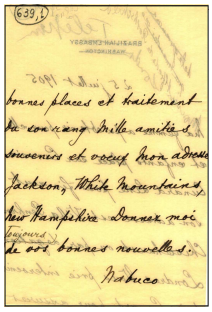
Sixteen images with different types of back-to-front interference were used in this study. Fourteen of the images are representative of a wide variety of historical documents and belong to the bequest of documents of Joaquim Nabuco shown in Figure 3.5 (a) to (n) held by the Joaquim Nabuco Foundation FUNDAJ (2016), a social science research centre in Recife, Brazil. Two other images, shown in Figure 3.6 (a) and (b) were from the Live Memory project, Lins (2010a), and presents the variety of printed documents of the proceedings of the conferences of the SBT (Sociedade Brasileira de Telecomunicacões).



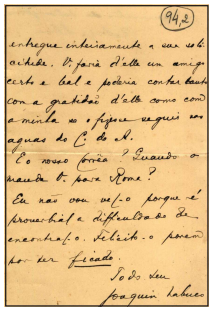
(a) Historical Document 1.



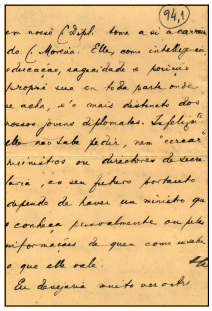
(b) Historical Document 2.



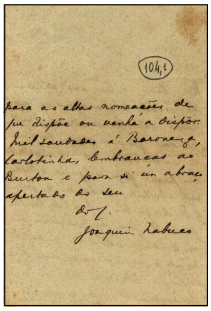
(c) Historical Document 3.



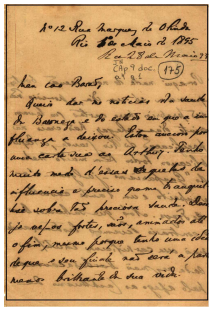
(d) Historical Document 4.



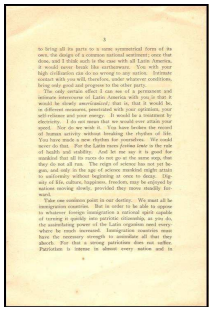
(e) Historical Document 5.



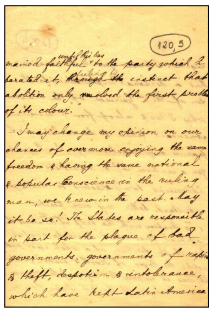
(f) Historical Document 6.



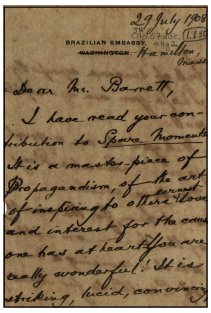
(g) Historical Document 7.



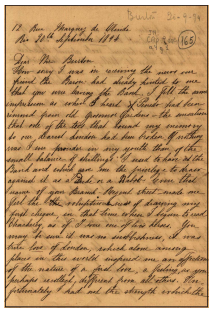
(h) Historical Document 8.



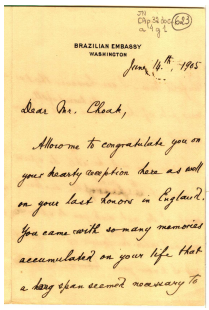
(i) Historical Document 9.



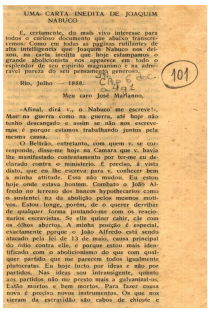
(j) Historical Document 10.



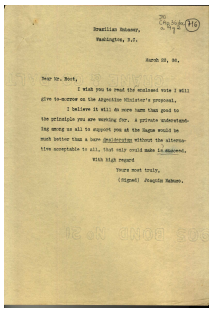
(k) Historical Document 11.



(l) Historical Document 12.



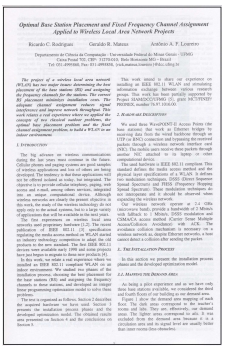
(m) Historical Document 13.



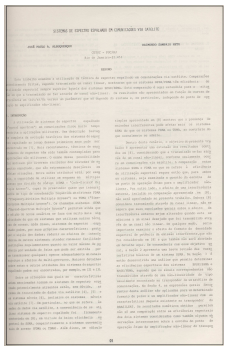
(n) Historical Document 14.

Figure 3.5: Historical Documents from Nabuco's bequest.

In a document such as the one presented in Figure 3.1, one expects to find three color clusters corresponding to the ink in the foreground, the paper background and



(a) Printed Document 15.



(b) Printed Document 16.

Figure 3.6: Printed Documents from the SBrT file.

the bleeding ink (the back-to-front interference), (Silva, 2006), but their distributions overlap as already showed in Figure 3.2. Although the technical literature presents several algorithms to filter and the back-to-front interference no algorithm is an "all case winner" as the strength of the interference may vary widely dependency on a number of factors such as the translucidity of the paper, its porosity, the kind of the ink used (water or oil based), the color of ink, the age of document, the conditions of document storage, etc. The goal here is to create a controlled environment, to generate synthesized images to filter them using the most successful algorithms in the literature to be able to detect the parameters in which their performance and their efficiency reach then the best. The block diagram in Figure 3.7 sketches the environment developed.

The specification and analysis of a system that generates a synthetic image of historical documents is the main goal of this implementation.

These operations were implemented by block diagram represented in Figure 3.7. The method used for generating synthesized documents followed reference Lins (2010b):

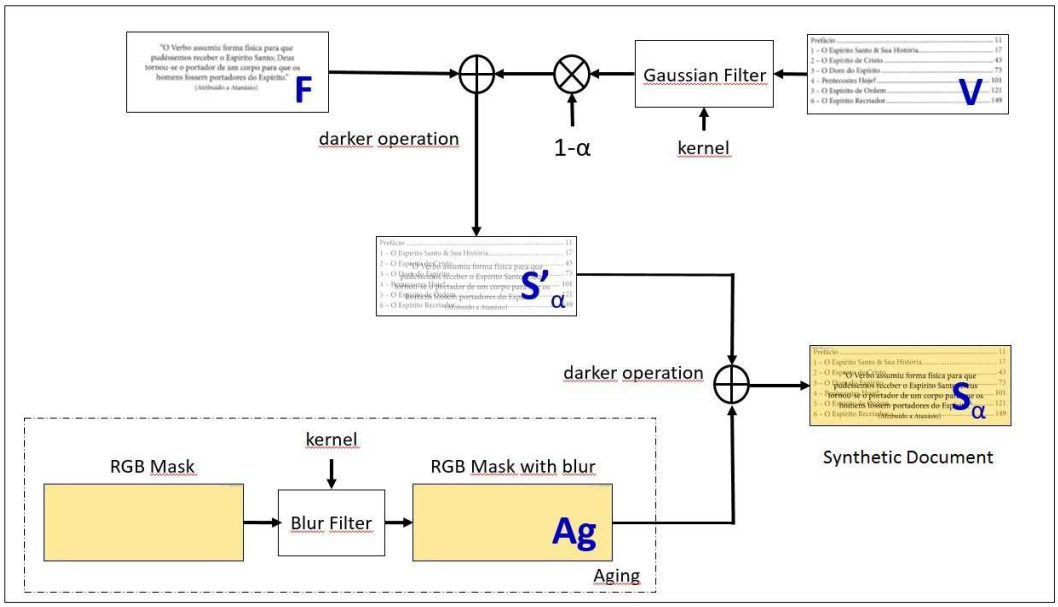
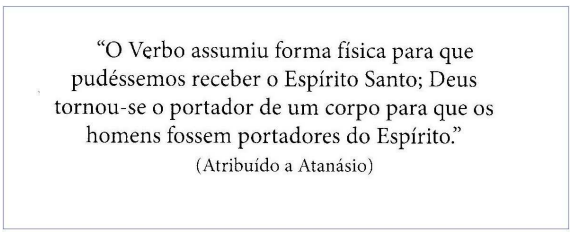
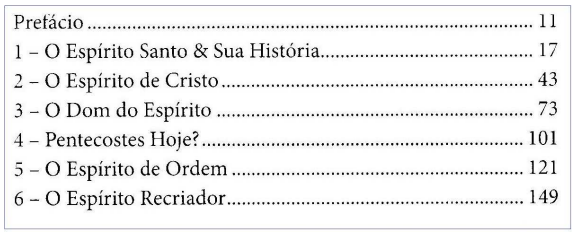


Figure 3.7: Block Diagram of the synthetic image generator.

Step 1 - Initially, two monochromatic document images without back-to-front interference, both images with 1200×480 pixels and a resolution of 300 dpi, such as the ones in Figure 3.8 are used.



(a) Front Image - F



(b) Back Image - V

Figure 3.8: Original images without back-to-front interference.

Step 2 - A third image S'_α is generated by composing the images F and V.

Linear spatial filtering modifies the image f by replacing the value at each pixel with some linear function of the values of nearby pixels. A spatial filter is an image operation where each pixel value $I(u;v)$ is changed by a function of the intensities of pixels in the neighborhood of $(u;v)$. Figure 3.9 shows how a linear spatial filter works. The filters described in Section 3.3 were applied to each synthetically

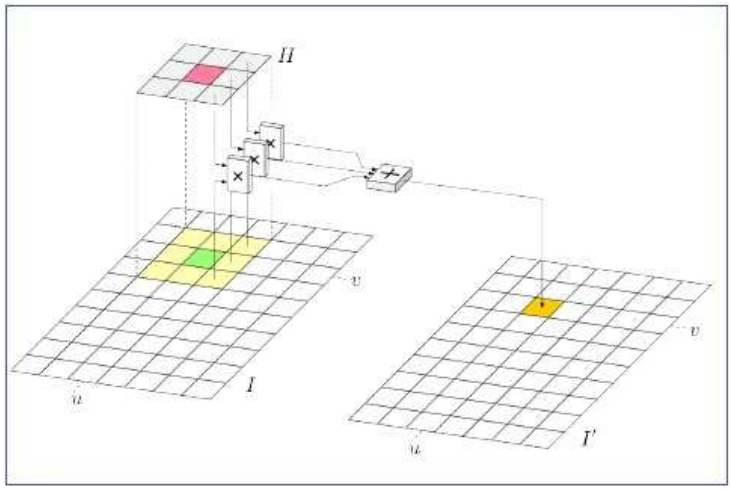


Figure 3.9: How a Linear Spatial Filter Works.

generated image with an old RGB value, with an alpha opacity factor used as input.

To model the unsharpness of the text from the back side when seen from the front side, a blur filter that is a kind of linear filter is applied to the reverse image, then, the image V (Back or Verse) was multiplied by the factor $(1-\alpha)$, where the index α represents the opacity coefficient of the image that assumes values from 0 to 100%. Otherwise, $(1-\alpha)$ indicates the transparency coefficient. The new components of this image are given by:

$$S'_\alpha = F \bigoplus (1-\alpha)V, \tag{3.2.1}$$

The resulting image shown in Figure 3.10 contains an example of image of document with back-to-front interference for $\alpha = 0.5$.

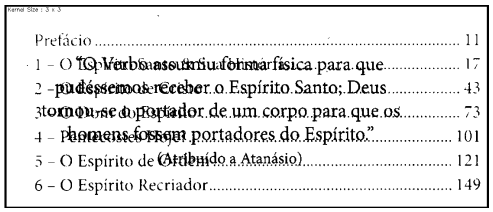


Figure 3.10: Image of document with back-to-front interference.

Step 3 - For each of the sixteen document images in this study a sample of the color of the aged paper background, shown in Figure 3.1, was measured for mean, variance and grayscale intensity of RGB, as texture shown in Figure 3.11.

A processing environment was envisaged to extract the basic features from each document. The extraction of these features was performed with the blocks, as presented in Figure 3.11.

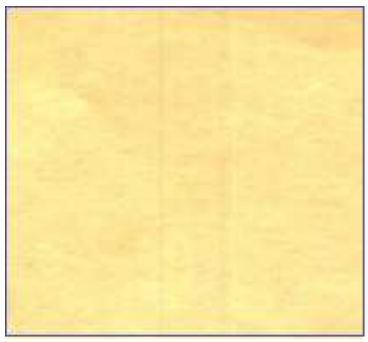
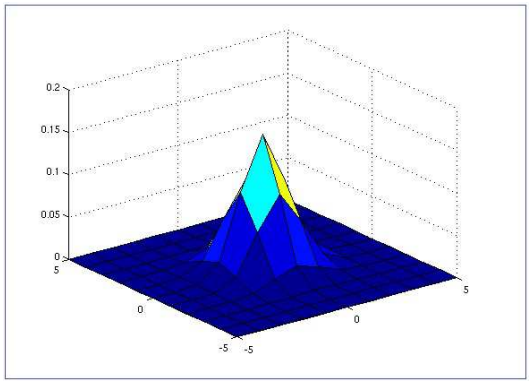
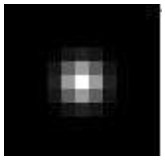


Figure 3.11: Sample texture of historical document.

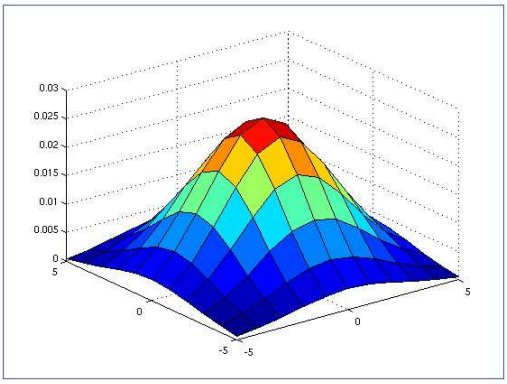
The effect of a Gaussian filter on the block diagram is showed as an example for $\sigma = 1$ and $\sigma = 2.5$ as shown in Figure 3.12.



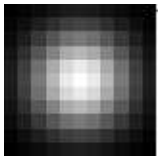
(a) Gaussian distribution for $\sigma = 1$



(b) For kernel of the size 11



(c) Gaussian distribution for $\sigma = 2.5$



(d) For kernel of the size 11

Figure 3.12: Example of Gaussian distribution for $\sigma = 1$ and $\sigma = 2.5$ and corresponding Gaussian size 11 kernels.

Step 4 - A sample of this document was used to measure the average and variance values of the RGB texture, using RGB measure, ImageJ application.

Table 3.1 shows the results of this evaluation, the value of the mean and standard deviation of the RGB components which contain all the information needed to create a synthetic image of the texture of the paper. Mello & Lins (2002) used the similar parameters to generate color synthetic images in a compression scheme. These values feed the blur filter kernel for aging of the paper in the block diagram as shown in Figure 3.7. The grayscale intensity is calculated by Equation 3.1.1.

Table 3.1: Texture information of historical document.

Label	Mean	Standard Deviation	Mode	Min	Max
Red	252.740	2.506	255	226	255
Green	233.209	6.252	234	193	248
Blue	153.654	4.907	155	126	167
GrayScale Intensity	229.99	4.65	231	196	241

The paper background image generated after applied the aging information for presented in the Table 3.1, it is shown in Figure 3.13.

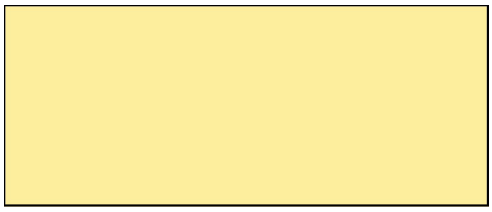


Figure 3.13: Aging mask generated by image synthesizer.

Step 5 - The image with back-to-front interference thus supplied the information for aging.

Finally, a "darker operation" is performed between the S' and A_g images, generating the final image S_α, according Equation 3.2.2. This operation is done pixel by pixel, comparing the luminance of the correspondent pixels on both images, as represented in Figure 3.14.

$$S_\alpha = S'_\alpha \bigoplus A_g. \tag{3.2.2}$$

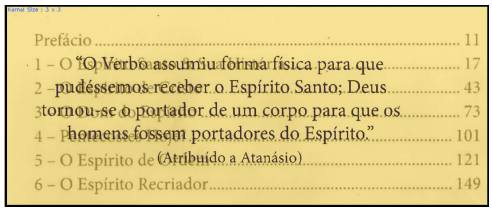


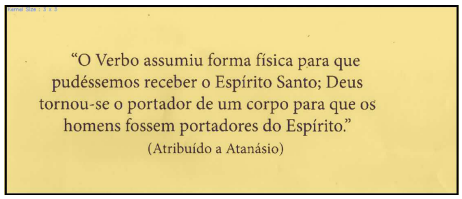
Figure 3.14: Synthesized image with back-to-front interference generated by the scheme in Figure 3.7.

3.2.1 Removing the Back-to-Front Interference

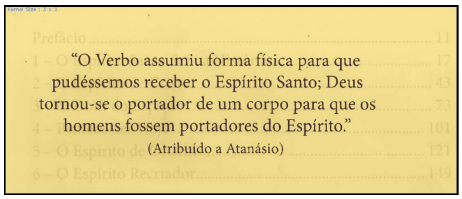
This section presents how test images with back-to-front interference were generated. The basic idea is to introduce such interference in a well controlled way, thus one is able to really know which pixels ought to be removed and which should not be removed in the filtered image. The mismatching pixels from the reference and filtered images was used to calculate quality factors of the algorithm, allowing a fair comparison between the results obtained.

Ten documents similar to the one shown in Figure 3.15 were created using the scheme in Figure 3.7 with exhibition of the parameters found in the historical file of letters by Joaquim Nabuco. The opacity coefficient α, varying between 0.1 to 1 in steps by 0.1, representing different strength of the back-to-front interference in the document.

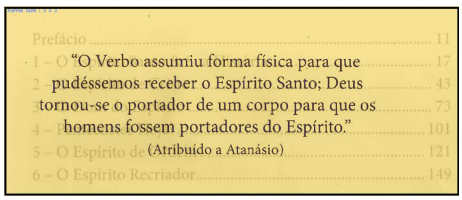
The blur - defocusing was modeled as a Gaussian filter, representing the point spread function, with standard error of blur in units of output pixel size. The blur size was chosen, in pixel, as 3×3, simulating a realistic aging effect in a document.



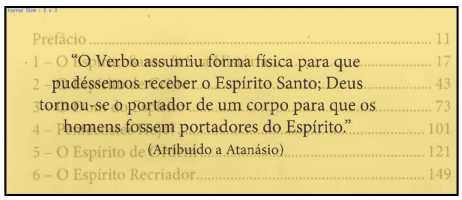
(a) Synthetic Image with $\alpha = 1$



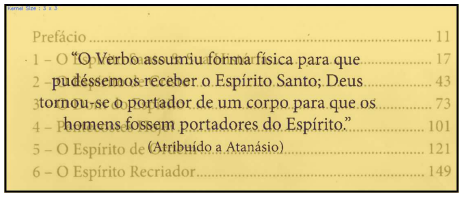
(b) Synthetic Image with $\alpha = 0.9$



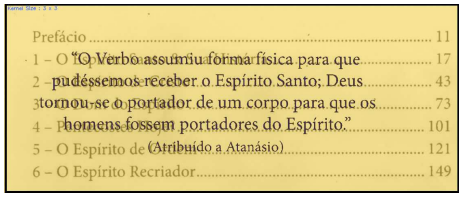
(c) Synthetic Image with $\alpha = 0.8$



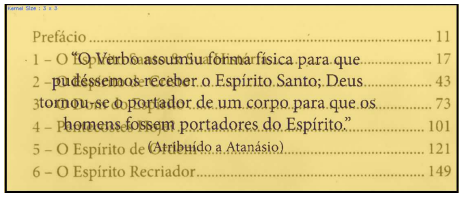
(d) Synthetic Image with $\alpha = 0.7$



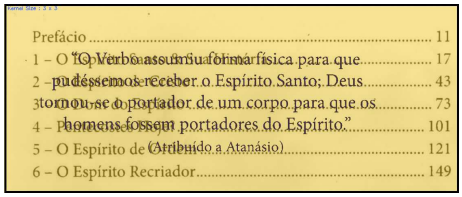
(e) Synthetic Image with $\alpha = 0.6$



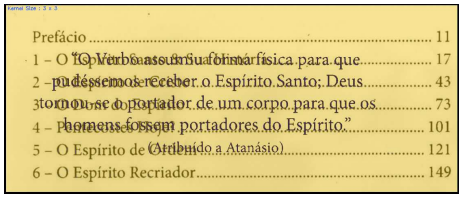
(f) Synthetic Image with $\alpha = 0.5$



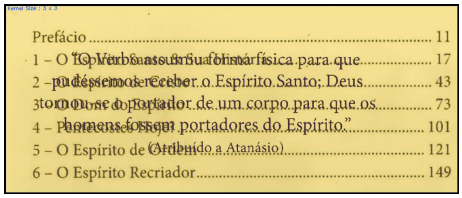
(g) Synthetic Image with $\alpha = 0.4$



(h) Synthetic Image with $\alpha = 0.3$



(i) Synthetic Image with $\alpha = 0.2$



(j) Synthetic Image with $\alpha = 0.1$

Figure 3.15: Synthetic images generated.

The 160 generated images have 1200×480 pixels and resolution of 300 dpi and 24 bits RGB. Mean and standard deviation values are shown on Table 3.1. The block diagram shown in Figure 3.7 was implemented and simulated in C^{++} using Visual Studio Platform with "OpenCV" libraries. The program developed in C^{++} is presented in Appendix C.

An example of the generated synthetic images are shown in Figure 3.15.

3.2.2 Co-Occurrence Matrices as a Measure of Quality

This Section analyzes various types of filtering techniques to remove the back-to-front interference, after the images shown in Figure 3.15 were transformed into grayscale. The techniques used were: IsoData, Pun, Kapoo-Sahoo-Wong, Johannsen-Bille, Yen-Chang-Chang, Otsu, Mello-Lins, Silva-Lins Rocha and Wu-Lu methods, where the filter input synthesized images by the process described in Figure 3.7. The output image was binarized and the threshold was calculated from each algorithm. The measure to evaluate the quality of the binarized images makes use of the matrix of co-occurrence probabilities.

In the binarization process after filtering, the co-occurrence probabilities of the original image and of the binary image were calculated. More specifically, the four quadrants of the co-occurrence matrix were considered, as illustrated in Figure 3.16. There the first quadrant denotes the number of pixels background-to-background (b, b), and the correct mapping of background pixels from the original image that correspond to background pixels in the filtered one. Similarly, the third quadrant stands for the foreground-to-foreground (f, f) correct mappings. The second and fourth quadrants denote, respectively, the background-to-foreground (b, f) and the foreground-to-background (f, b) uncorrect transformations. Using the co-occurrence probabilities $p_{i,j}$, that is, the score of i to j gray-

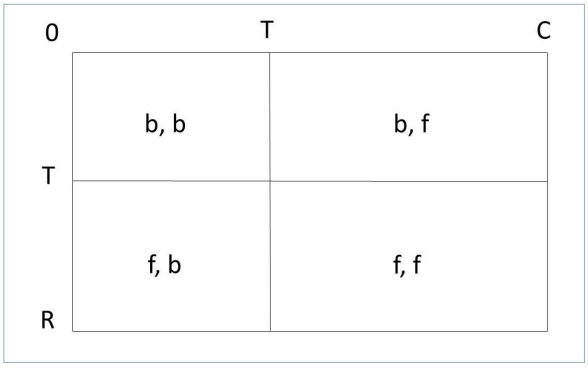


Figure 3.16: Co-occurrence matrix.

level transitions normalized by the total number of transitions, the quadrant probabilities were as:

$$P_{bb}(T) = \sum_{i=0}^{T} \sum_{j=0}^{T} p_{ij}. \tag{3.2.3}$$

$$P_{bf}(T) = \sum_{i=0}^{T} \sum_{j=T+1}^{C} p_{ij}. \tag{3.2.4}$$

$$P_{fb}(T) = \sum_{i=T+1}^{R} \sum_{j=0}^{T} p_{ij}. \tag{3.2.5}$$

$$P_{ff}(T) = \sum_{i=T+1}^{R} \sum_{j=T+1}^{C} p_{ij}. \tag{3.2.6}$$

In conclusion, the performance underwent a comparison among the various types of filters.

The procedure performed on filter analysis is shown in Figure 3.17, and was implemented and simulated using ImageJ plugins. The several synthetic images went through

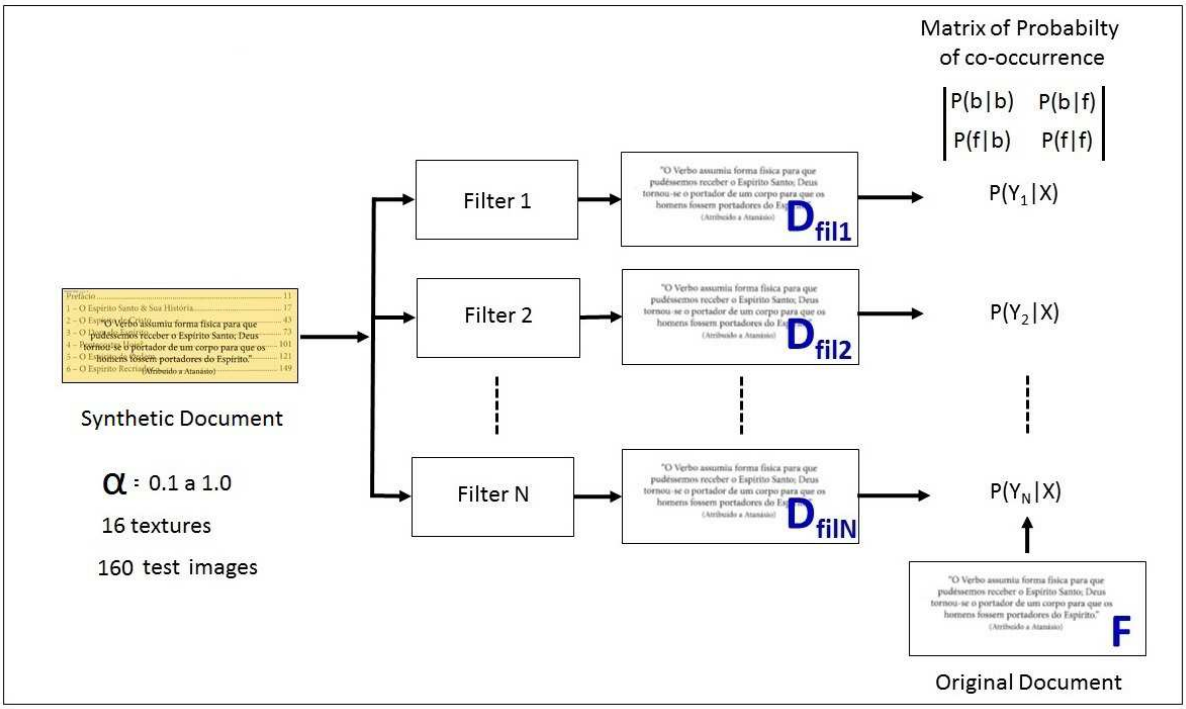


Figure 3.17: Procedures for the analysis of filters.

the different binarization filters, and the output image of each filter. Then, the matrix of co-occurrence probability - $P(Y|X)$, where X was the input and Y the output images, is calculated as represented in Equation 3.2.7.

$$P(Y|X) = \begin{matrix} \\ b \\ f \end{matrix} \begin{matrix} \begin{matrix} b & \quad f \end{matrix} \\ \begin{pmatrix} P(b|b) & P(b|f) \\ P(f|b) & P(f|f) \end{pmatrix} \end{matrix} \tag{3.2.7}$$

3.3 Thresholding Algorithms

Thresholding algorithms can be classified into *global* or *local* algorithms. Global algorithms define a unique threshold value for the complete image. Local algorithms first split the image into regions according to some criterion and then define threshold values for each region. In general, global algorithms are faster than local algorithms, although, in general, local algorithms may provide better results. Otsu is the most widely used global thresholding algorithm. Otsu's algorithm is adaptive and requires no adjustment setting. It considers that there are two classes, separated by a threshold value. The inter and intra class variances are evaluated and the final threshold value is the one that maximizes the relation between them.

Global thresholding based on clustering (Otsu, 1979), entropy minimization (Kapur *et al.*, 1985), and valley seeking in the intensity histogram (Weszka, 1979) as well as feature-based (Dawoud & Kamel, 2004) and model-based (Liu & Srihari, 1997) methods have been proposed. These methods are efficient for images in which the gray levels of the text and background pixels are separable. If the histogram of the text overlaps with the one of the background, the result are poor quality binary images, (Valizadeh & Kabir, 2012).

Sezgin & Sankur (2004) presented a comprehensive overview and comparison of thresholding algorithms, clustering them according to their nature. From the almost forty algorithms presented six schemes were deemed to be suitable to work in documents with back-to-front interference such as the ones studied here: Pun (1981), Kapur *et al.* (1985), Johannsen & Bille (1982), Yen (1995), Wu-Songde-Hanquing (1998), and Otsu (1979). The first five algorithms are based on the entropy of the image, whereas the last one uses discriminator analysis. Iterative Self Organizing Data Analysis Technique - IsoData was added. It is a method of unsupervised classification, and the computer runs the algorithm through many iterations until threshold is reached; all algorithms work with thresholding techniques. Appendix *A* outlines how such algorithms work.

Besides those algorithms, two algorithms based on Shannon's entropy, that were created in the scope of the Nabuco Project to filter that interference were also accessed: Mello & Lins (2002), Mello & Lins (2000) and Silva (2006).

The Appendix B of this thesis presents an updated annotated bibliography of the algorithms found in the literature for document image binarization techniques, including documents with back-to-front interference.

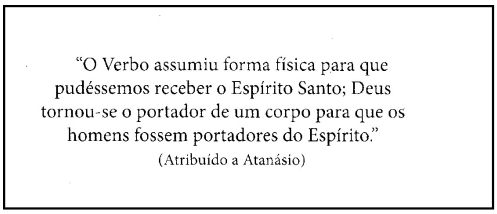
3.3.1 The IsoData Method

Unsupervised clustering is a fundamental tool in image processing for geoscience and remote sensing applications. For example, unsupervised clustering is often used to obtain vegetation maps of an area of interest according to (Memarsadeghi, 2007).

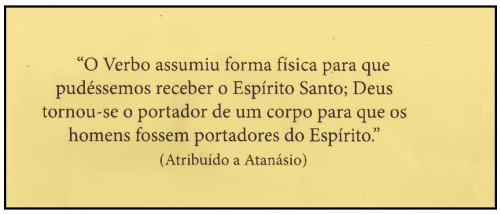
Most common unsupervised image classification is Isodata, Support Vector Machine (SVM) and k-Means methods, according to (Abburu & Golla, 2015).

Figure 3.18 (a) shows a standard image without front-to-back interference, which will be compared to the binarized image at the filter output, whereas Figure 3.18 (b) shows the colored synthetic image to be filtered.

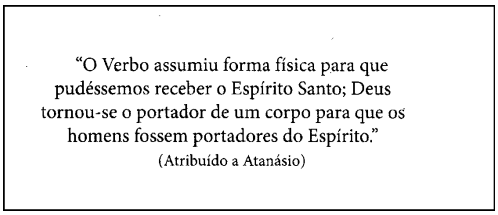
The result of applying IsoData Filter using the document images of Figure 3.15 is shown in Figure 3.18.



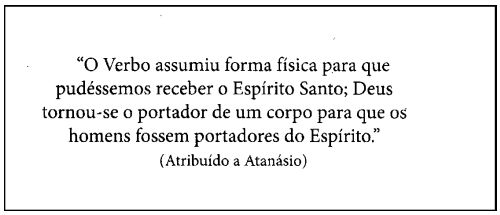
(a) Original Input Image



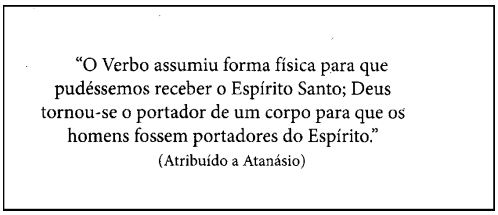
(b) Synthetic Image to be filtered



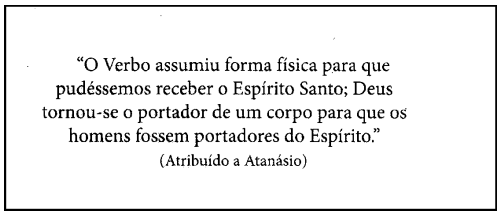
(c) Filtered Image with $\alpha = 1$



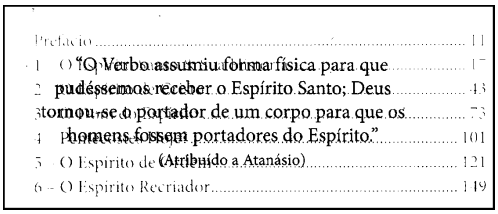
(d) Filtered Image with $\alpha = 0.9$



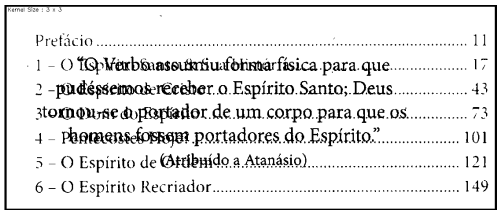
(e) Filtered Image with $\alpha = 0.8$



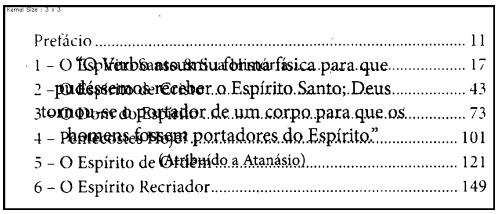
(f) Filtered Image with $\alpha = 0.7$



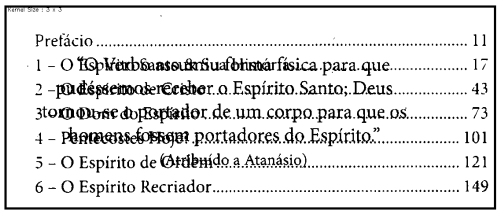
(g) Filtered Image with $\alpha = 0.6$



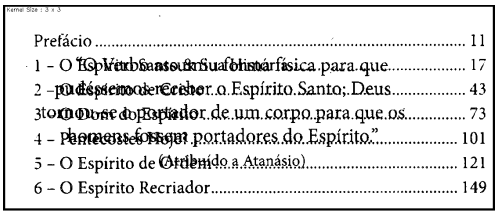
(h) Filtered Image with $\alpha = 0.5$



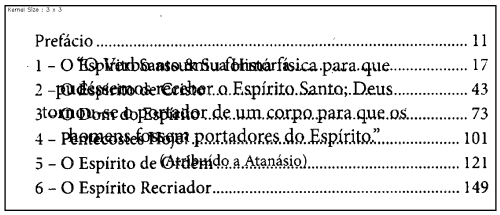
(i) Filtered Image with $\alpha = 0.4$



(j) Filtered Image with $\alpha = 0.3$



(k) Filtered Image with $\alpha = 0.2$



(l) Filtered Image with $\alpha = 0.1$

Figure 3.18: ISODATA filtering of the images in Figure 3.15.

Table 3.2 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

Table 3.2: IsoData Filter Result.

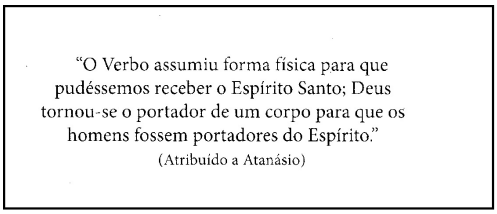
α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	142	94.49%	5.51%	99.52%	0.48%
0.2	1x1	153	233	252	3x3	142	94.84%	5.16%	99.53%	0.47%
0.3	1x1	153	233	252	3x3	144	95.14%	4.86%	100.00%	0.00%
0.4	1x1	153	233	252	3x3	146	95.54%	4.46%	100.00%	0.00%
0.5	1x1	153	233	252	3x3	147	96.22%	3.78%	100.00%	0.00%
0.6	1x1	153	233	252	3x3	144	97.85%	2.15%	100.00%	0.00%
0.7	1x1	153	233	252	3x3	136	99.89%	0.11%	98.87%	1.13%
0.8	1x1	153	233	252	3x3	137	99.94%	0.06%	99.23%	0.77%
0.9	1x1	153	233	252	3x3	137	99.98%	0.02%	99.20%	0.80%
1.0	1x1	153	233	252	3x3	138	100.00%	0.00%	99.56%	0.44%

Analyzing the quality of the binarized images produced by the Isodata filter in Figure 3.18 (c) to (f), it seems reasonable to consider important features for removing back-to-front interference: in the range, where the α varied between 1.0 to 0.7, the value of background-background probability $P(b|b)$ varied between 100.00% and 99.89%, respectively, a error less than 0.11%, considering that the co-occurrence probability $P(f|f)$ had a small variation between 99.56% and 98.87%, a error less than 1.13%. While from the images in Figure 3.18 (g) to (l) the back-to-front interference is present, the proportion of an α reduction.

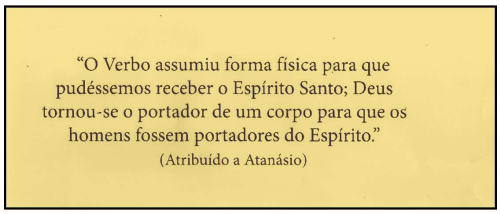
3.3.2 Pun Method

In the algorithm proposed by Pun (1981), the gray levels are considered as produced by a source with an alphabet consisting of 256 statistically independent symbols. Pun considered the ratio between the *a posteriori* entropy and the total entropy.

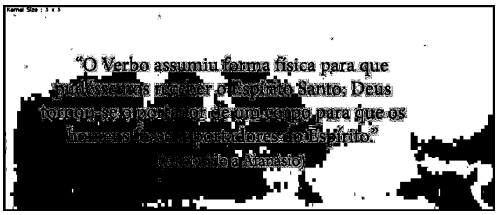
Figure 3.19 presents the result of applying Pun's algorithm from the generated synthetic images presented in Figure 3.15.



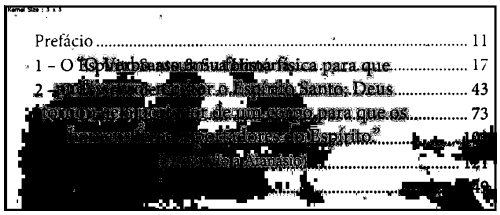
(a) Original Input Image



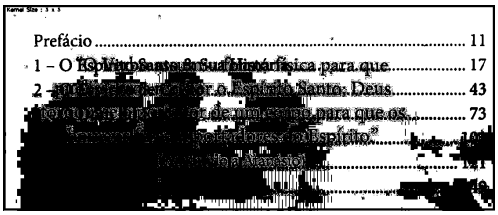
(b) Synthetic Image to be filtered



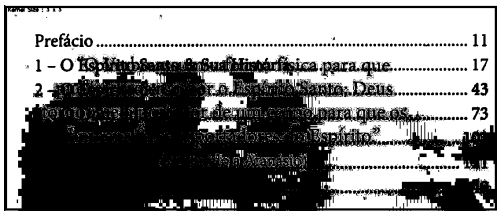
(c) Filtered Image with $\alpha = 1$



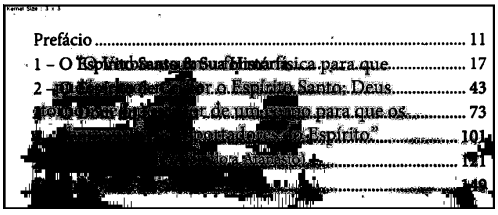
(d) Filtered Image with $\alpha = 0.9$



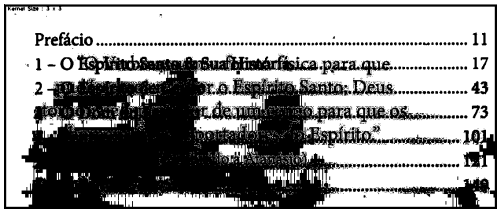
(e) Filtered Image with $\alpha = 0.8$



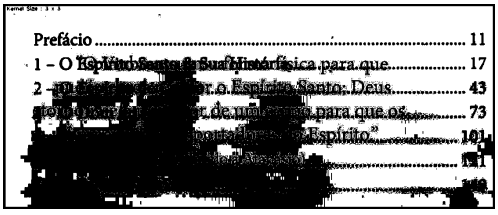
(f) Filtered Image with $\alpha = 0.7$



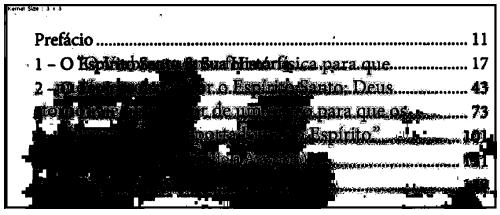
(g) Filtered Image with $\alpha = 0.6$



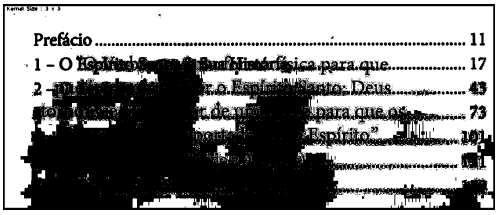
(h) Filtered Image with $\alpha = 0.5$



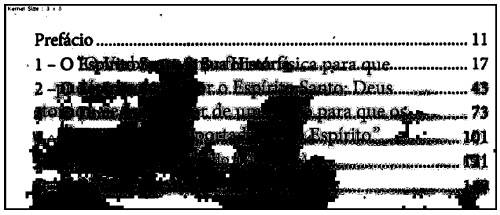
(i) Filtered Image with $\alpha = 0.4$



(j) Filtered Image with $\alpha = 0.3$



(k) Filtered Image with $\alpha = 0.2$



(l) Filtered Image with $\alpha = 0.1$

Figure 3.19: Pun filtering of the images in Figure 3.15.

Table 3.3 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

Table 3.3: Pun Filter Result.

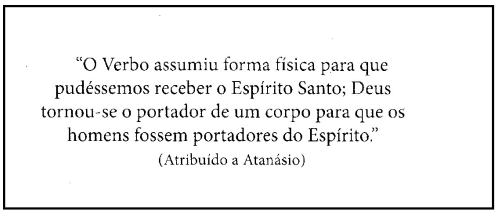
(α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	195	61.99%	38.01%	100.00%	0.00%
0.2	1x1	153	233	252	3x3	196	57.97%	42.03%	100.00%	0.00%
0.3	1x1	153	233	252	3x3	196	59.15%	40.85%	100.00%	0.00%
0.4	1x1	153	233	252	3x3	196	61.64%	38.36%	100.00%	0.00%
0.5	1x1	153	233	252	3x3	196	65.20%	34.80%	100.00%	0.00%
0.6	1x1	153	233	252	3x3	196	67.16%	32.84%	100.00%	0.00%
0.7	1x1	153	233	252	3x3	198	55.51%	44.49%	100.00%	0.00%
0.8	1x1	153	233	252	3x3	198	58.39%	41.61%	100.00%	0.00%
0.9	1x1	153	233	252	3x3	198	60.52%	39.48%	100.00%	0.00%
1.0	1x1	153	233	252	3x3	199	59.76%	40.24%	100.00%	0.00%

A closer look at the binarized images in Figure 3.19, yields to observe that Pun's algorithm did not produce good quality binarized images. Besides, the minimum value of the background-background probability $P(b|b)$ presented in Table 3.3 was 55.51%, an error of 44.49, considering that the foreground-foreground probability $P(f|f)$ was of 100.00%, for an $\alpha = 0.7$.

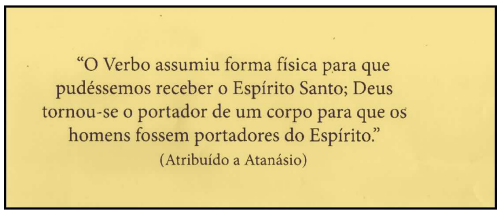
3.3.3 Kapur-Sahoo-Wong Filter

The algorithm by Kapur *et al.* (1985) considers the foreground and background images as two distinct sources, such that whenever the addition of the two entropies reach a maximum, its argument t reaches the optimal value.

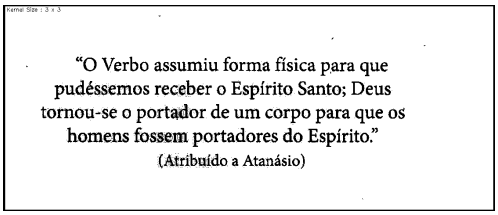
Figure 3.20 shows the result of applying Kapur-Sahoo-Wong Filter from the synthetically generated images showed in Figure 3.15.



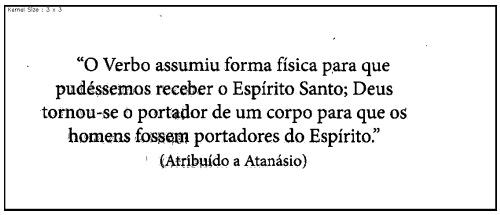
(a) Original Input Image



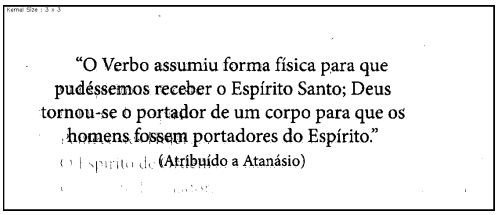
(b) Synthetic Image to be filtered



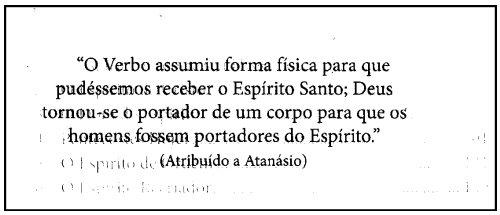
(c) Filtered Image with $\alpha = 1$



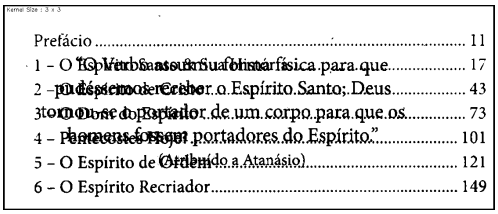
(d) Filtered Image with $\alpha = 0.9$



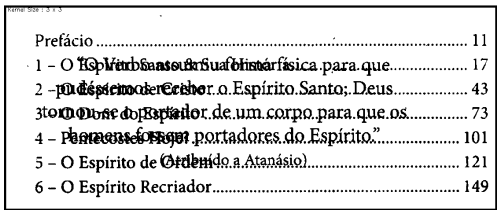
(e) Filtered Image with $\alpha = 0.8$



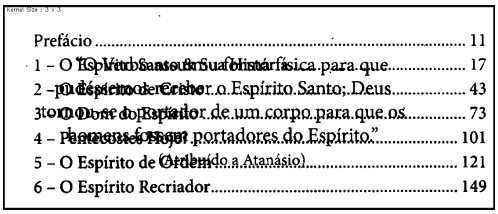
(f) Filtered Image with $\alpha = 0.7$



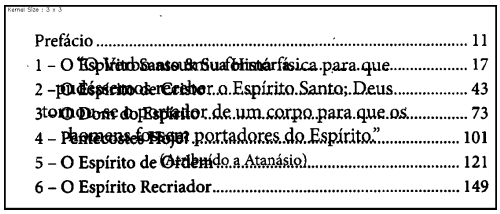
(g) Filtered Image with $\alpha = 0.6$



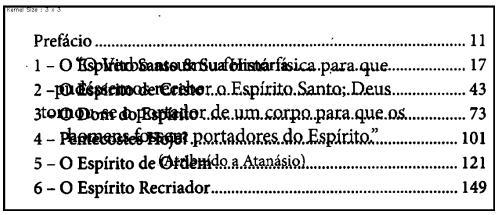
(h) Filtered Image with $\alpha = 0.5$



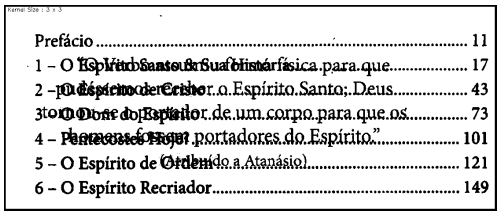
(i) Filtered Image with $\alpha = 0.4$



(j) Filtered Image with $\alpha = 0.3$



(k) Filtered Image with $\alpha = 0.2$



(l) Filtered Image with $\alpha = 0.1$

Figure 3.20: Kapur-Sahoo-Wong filtering of the images in Figure 3.15.

Table 3.4 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

Table 3.4: Kapur-Sahoo-Wong Filter Result.

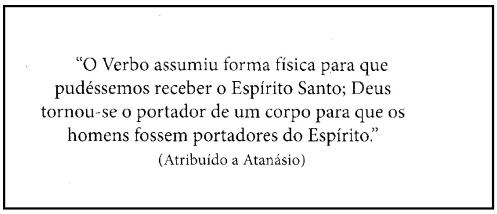
α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	176	90.88%	9.12%	100.00%	0.00%
0.2	1x1	153	233	252	3x3	174	91.50%	8.50%	100.00%	0.00%
0.3	1x1	153	233	252	3x3	174	91.86%	8.15%	100.00%	0.00%
0.4	1x1	153	233	252	3x3	174	92.29%	7.71%	100.00%	0.00%
0.5	1x1	153	233	252	3x3	173	92.98%	7.02%	100.00%	0.00%
0.6	1x1	153	233	252	3x3	174	93.49%	6.51%	100.00%	0.00%
0.7	1x1	153	233	252	3x3	147	99.25%	0.75%	100.00%	0.00%
0.8	1x1	153	233	252	3x3	162	98.87%	1.13%	100.00%	0.00%
0.9	1x1	153	233	252	3x3	175	98.59%	1.41%	100.00%	0.00%
1.0	1x1	153	233	252	3x3	182	98.36%	1.64%	100.00%	0.00%

A analysis of the images in Figure 3.20 (c) to (f) revealed that there was the partial elimination the back-to-front interference, mainly figures (e) and (f), which correspond in the range where the alpha varied between 1.0 and 0.7, the value of background-background probability $P(b|b)$ varied between 99.25% and 98.36%, a error less than 1.64%, considering that the foreground-foreground probability $P(b|b)$ was of 100.00%.

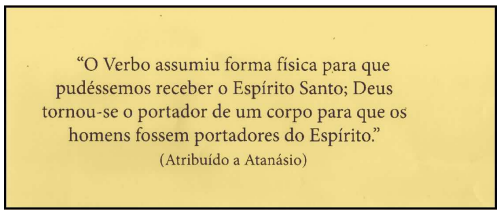
3.3.4 Johannsen-Bille Method

This method uses the entropy of the gray level histogram of the digital image. Essentially, it divides the set of gray into two parts, so as to minimize the interdependence between them.

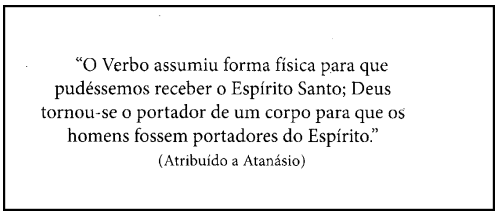
Figure 3.21 shows the result of applying Johannsen-Bille Method from the synthetically generated images showed in Figure 3.15.



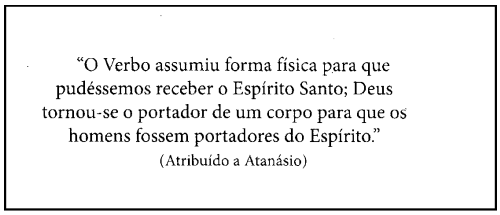
(a) Original Input Image



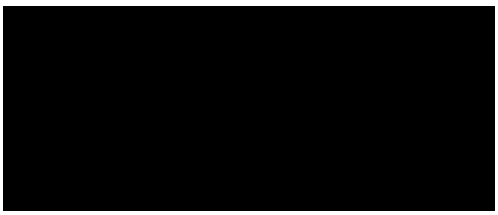
(b) Synthetic Image to be filtered



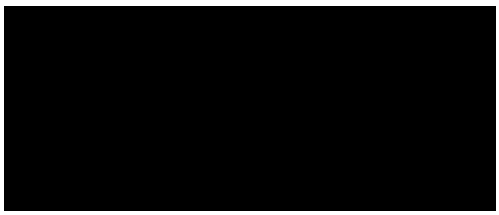
(c) Filtered Image with $\alpha = 1$



(d) Filtered Image with $\alpha = 0.9$

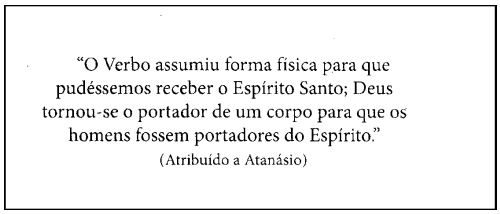


(e) Filtered Image with $\alpha = 0.8$

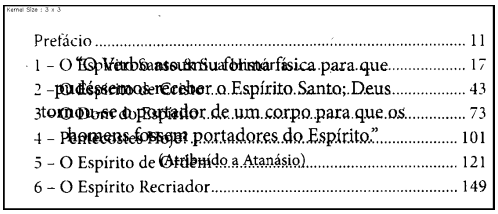


(f) Filtered Image with $\alpha = 0.7$

(g) Filtered Image with $\alpha = 0.6$

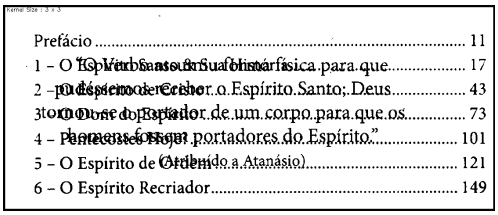


(h) Filtered Image with $\alpha = 0.5$

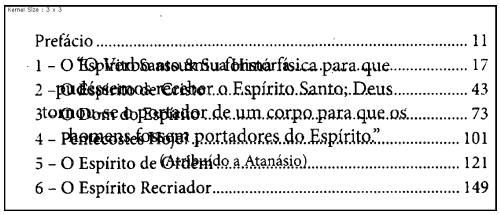


(i) Filtered Image with $\alpha = 0.4$

(j) Filtered Image with $\alpha = 0.3$



(k) Filtered Image with $\alpha = 0.2$



(l) Filtered Image with $\alpha = 0.1$

Figure 3.21: Johannsen-Bille filtering of the images in Figure 3.15.

Table 3.5 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

Table 3.5: Johannsen-Bille Filter Result.

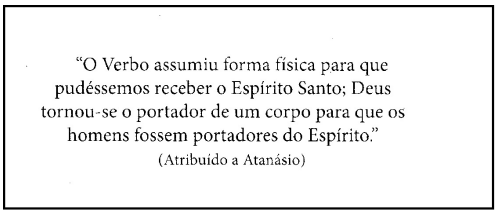
α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	142	94.49%	5.51%	99.52%	0.48%
0.2	1x1	153	233	252	3x3	149	94.23%	5.77%	100.00%	0.00%
0.3	1x1	153	233	252	3x3	210	0.00%	100.00%	100.00%	0.00%
0.4	1x1	153	233	252	3x3	150	95.15%	4.85%	100.00%	0.00%
0.5	1x1	153	233	252	3x3	100	99.97%	0.03%	84.63%	15.37%
0.6	1x1	153	233	252	3x3	211	0.00%	100.00%	100.00%	0.00%
0.7	1x1	153	233	252	3x3	211	0.00%	100.00%	100.00%	0.00%
0.8	1x1	153	233	252	3x3	211	0.00%	100.00%	100.00%	0.00%
0.9	1x1	153	233	252	3x3	112	100.00%	0.00%	88.39%	11.61%
1.0	1x1	153	233	252	3x3	112	100.00%	0.00%	88.11%	11.89%

Figure 3.21 (e), (f), (g) and (j) shows the inadequacy of applying the Johannsen-Bille filter from the set of synthesised image presented in Figure 3.15, however, for the $\alpha = 1.0, 0.9$ and 0.5 values the Johannsen-Bille filter has a reasonable response. The value of background-background probability $P(b|b)$ varied between 100.00% and 99.97%, while the value of foreground-foreground probability changed between 88.11% and 84.63%, offering a loss in the quality of the text.

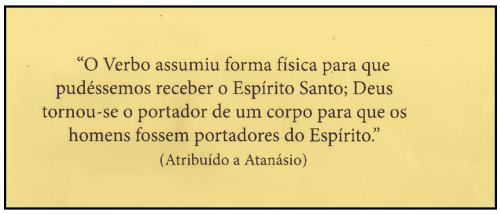
3.3.5 Yen-Chang-Chang Method

The algorithm by Yen (1995) follows the same idea as the one by Kapur *et al.* (1985) and his colleagues in respect to the entropy distributions.

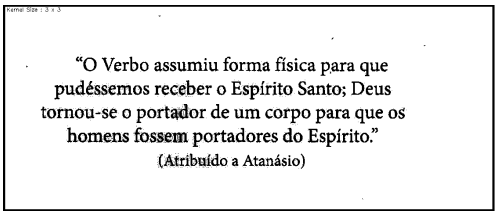
The result of applying Yen-Chang-Chang Method to the document image of Figure 3.15 is showed in Figure 3.22.



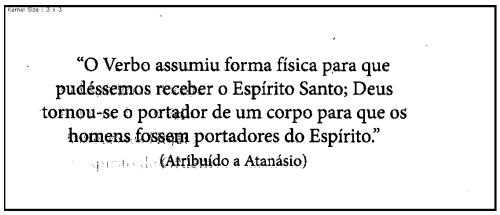
(a) Original Input Image



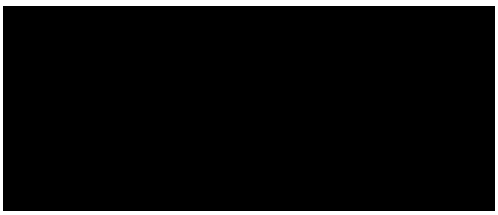
(b) Synthetic Image to be filtered



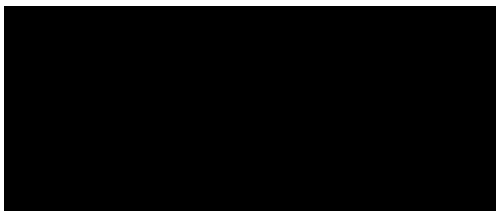
(c) Filtered Image with $\alpha = 1$



(d) Filtered Image with $\alpha = 0.9$

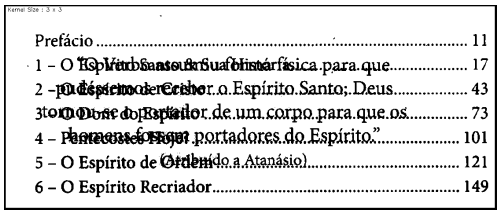


(e) Filtered Image with $\alpha = 0.8$



(f) Filtered Image with $\alpha = 0.7$

(g) Filtered Image with $\alpha = 0.6$



(h) Filtered Image with $\alpha = 0.5$

(i) Filtered Image with $\alpha = 0.4$

(j) Filtered Image with $\alpha = 0.3$

(k) Filtered Image with $\alpha = 0.2$

(l) Filtered Image with $\alpha = 0.1$

Figure 3.22: Yen-Chang-Chang filtering of the images in Figure 3.15.

Table 3.6 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

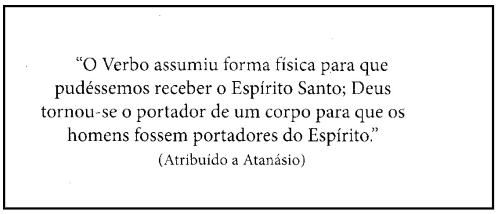
Table 3.6: Yen-Chang-Chang Filter Result.

α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	210	0.00%	100.00%	100.00%	0.00%
0.2	1x1	153	233	252	3x3	210	0.00%	100.00%	100.00%	0.00%
0.3	1x1	153	233	252	3x3	210	0.00%	100.00%	100.00%	0.00%
0.4	1x1	153	233	252	3x3	210	0.00%	100.00%	100.00%	0.00%
0.5	1x1	153	233	252	3x3	178	92.14%	7.86%	100.00%	0.00%
0.6	1x1	153	233	252	3x3	211	0.00%	100.00%	100.00%	0.00%
0.7	1x1	153	233	252	3x3	211	0.00%	100.00%	100.00%	0.00%
0.8	1x1	153	233	252	3x3	211	0.00%	100.00%	100.00%	0.00%
0.9	1x1	153	233	252	3x3	176	98.47%	1.53%	100.00%	0.00%
1.0	1x1	153	233	252	3x3	183	98.23%	1.77%	100.00%	0.00%

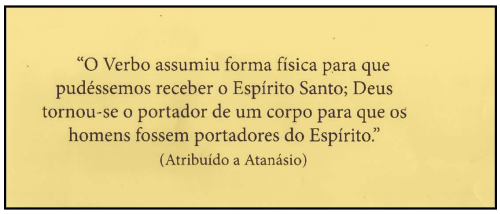
Figure 3.22 (e), (f), (g), (i), (j), (k) and (l) which corresponds to an $\alpha = 0.8, 0.7, 0.6, 0.4$, $0.3, 0.2$ and 0.1 value, respectively, the inadequacy of applying the Johannsen-Bille filter from the set of synthesised image presented in Figure 3.15, however, for the $\alpha = 1.0, 0.9$ and 0.5 values the Johannsen-Bille filter has a reasonable response. Only for $\alpha = 1$ and 0.9 the result of binarized image quality and the co-occurrence probability were acceptable.

3.3.6 Otsu Threshold Method

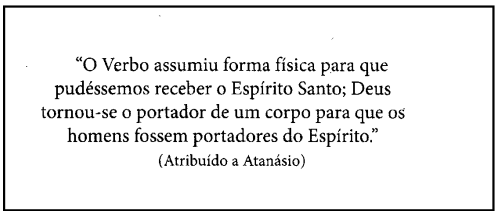
The algorithm by Otsu (1979) does not belong to the class of algorithms based on entropy. It is included here because it is one of the most often used algorithms in image segmentation. Otsu's algorithm makes a discriminator analysis, Sahoo (1988), for defining whether a gray level t will be mapped into object or background information. Figure 3.23 presents the result of the application of Otsu's algorithm to the image presented in Figure 3.15.



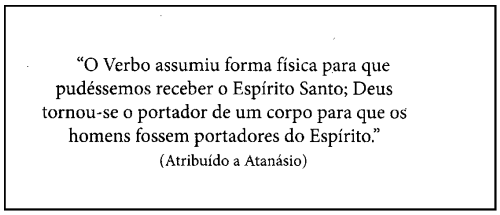
(a) Original Input Image



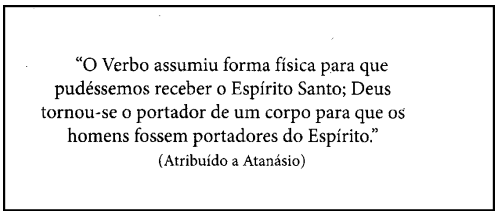
(b) Synthetic Image to be filtered



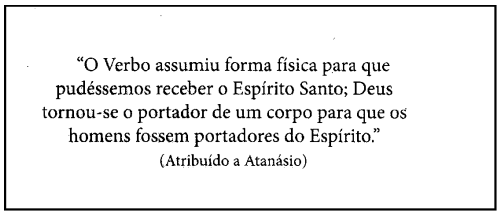
(c) Filtered Image with $\alpha = 1$



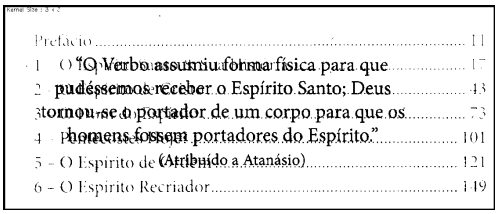
(d) Filtered Image with $\alpha = 0.9$



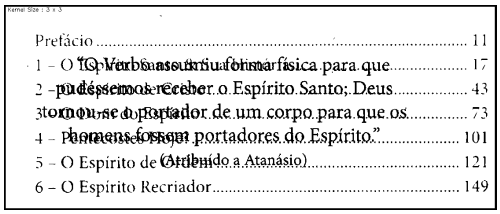
(e) Filtered Image with $\alpha = 0.8$



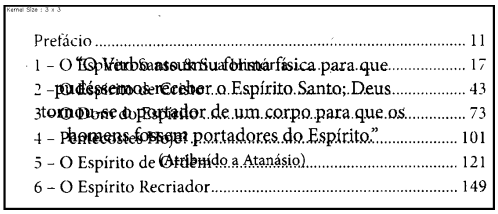
(f) Filtered Image with $\alpha = 0.7$



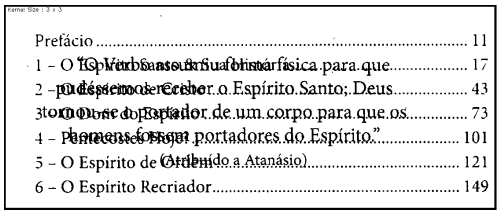
(g) Filtered Image with $\alpha = 0.6$



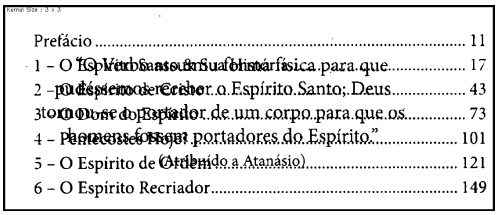
(h) Filtered Image with $\alpha = 0.5$



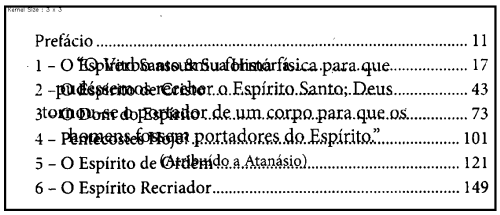
(i) Filtered Image with $\alpha = 0.4$



(j) Filtered Image with $\alpha = 0.3$



(k) Filtered Image with $\alpha = 0.2$



(l) Filtered Image with $\alpha = 0.1$

Figure 3.23: Otsu filtering of the images in Figure 3.15.

Table 3.7 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

Table 3.7: Otsu Filter Result.

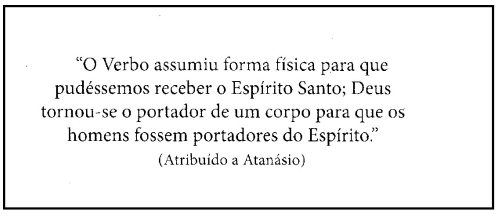
α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	145	94.19%	5.81%	100.00%	0.00%
0.2	1x1	153	233	252	3x3	145	94.57%	5.43%	100.00%	0.00%
0.3	1x1	153	233	252	3x3	145	95.05%	4.95%	100.00%	0.00%
0.4	1x1	153	233	252	3x3	149	95.24%	4.76%	100.00%	0.00%
0.5	1x1	153	233	252	3x3	149	96.00%	4.00%	100.00%	0.00%
0.6	1x1	153	233	252	3x3	146	97.51%	2.49%	100.00%	0.00%
0.7	1x1	153	233	252	3x3	138	99.87%	0.13%	99.54%	0.46%
0.8	1x1	153	233	252	3x3	138	99.94%	0.06%	99.56%	0.44%
0.9	1x1	153	233	252	3x3	138	99.97%	0.03%	99.53%	0.47%
1.0	1x1	153	233	252	3x3	140	99.95%	0.05%	99.55%	0.45%

For visual analysis of the binarized images in Figure 3.23 (c) to (f), was observed that when alpha varied between 1.0 to 0.7, the value of background-background probability $P(b|b)$ varied between 99.97% and 99.87%, a error less than 0.13%, considering that the foreground-foreground probability $P(f|f)$ varied of 99.56% and 99.53%, a error less than 0.47%. While from the images in Figure 3.23 (g) to (l) the back-to-front interference is present, the proportion of an alpha reduction.

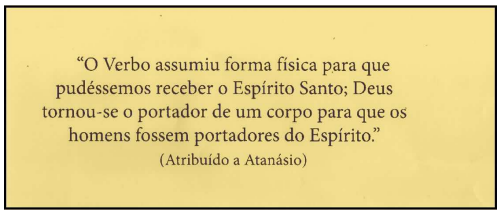
3.3.7 Mello-Lins Algorithm

The algorithm by Mello & Lins (2002) and Mello & Lins (2000) looks for the most frequent gray level of the image and takes it like initial threshold to evaluate the values H_b, H_w and H, shown in Appendix A.

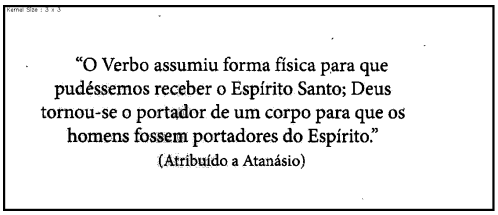
The result of applying Mello-Lins Algorithm to the document image of Figure 3.15 is showed in Figure 3.22.



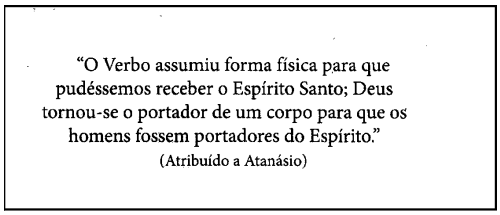
(a) Original Input Image



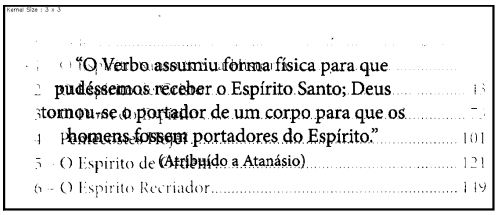
(b) Synthetic Image to be filtered



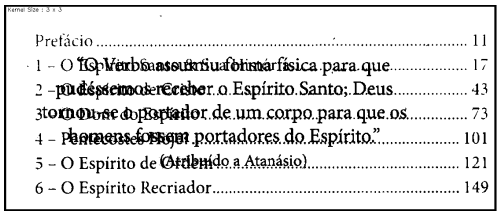
(c) Filtered Image with $\alpha = 1$



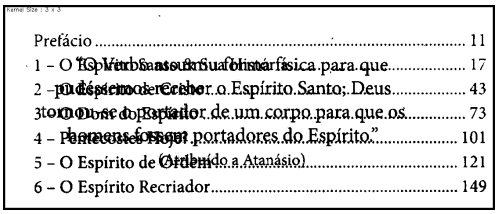
(d) Filtered Image with $\alpha = 0.9$



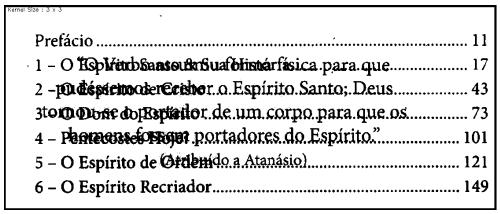
(e) Filtered Image with $\alpha = 0.8$



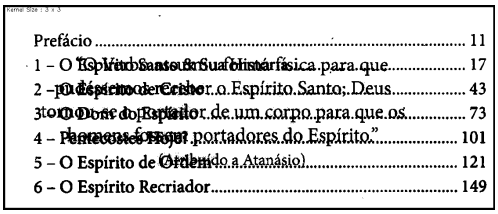
(f) Filtered Image with $\alpha = 0.7$



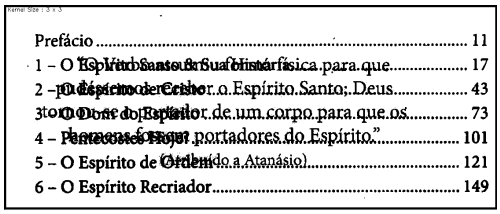
(g) Filtered Image with $\alpha = 0.6$



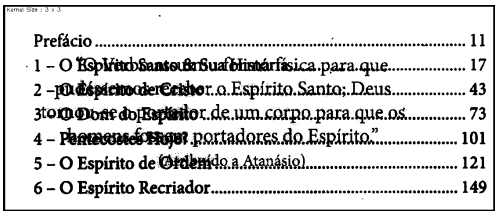
(h) Filtered Image with $\alpha = 0.5$



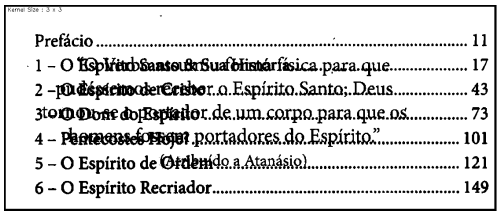
(i) Filtered Image with $\alpha = 0.4$



(j) Filtered Image with $\alpha = 0.3$



(k) Filtered Image with $\alpha = 0.2$



(l) Filtered Image with $\alpha = 0.1$

Figure 3.24: Mello-Lins filtering of the images in Figure 3.15.

Table 3.8 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

Table 3.8: Mello-Lins Filter Result.

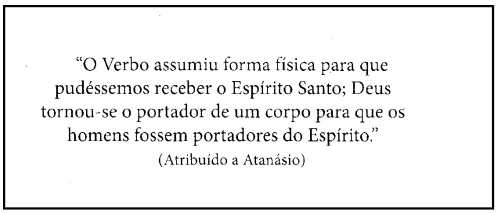
α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	174	91.19%	8.81%	100.00%	0.00%
0.2	1x1	153	233	252	3x3	183	89.76%	10.24%	100.00%	0.00%
0.3	1x1	153	233	252	3x3	181	90.58%	9.42%	100.00%	0.00%
0.4	1x1	153	233	252	3x3	180	91.21%	8.78%	100.00%	0.00%
0.5	1x1	153	233	252	3x3	178	92.14%	7.86%	100.00%	0.00%
0.6	1x1	153	233	252	3x3	176	93.14%	6.86%	100.00%	0.00%
0.7	1x1	153	233	252	3x3	174	94.47%	5.53%	100.00%	0.00%
0.8	1x1	153	233	252	3x3	170	97.30%	2.70%	100.00%	0.00%
0.9	1x1	153	233	252	3x3	165	99.19%	0.81%	100.00%	0.00%
1.0	1x1	153	233	252	3x3	181	98.45%	1.55%	100.00%	0.00%

A examination of the binarized images in Figure 3.24 (c) to (f) reveals that there was the partial elimination the back-to-from interference, mainly figures (e) and (f), which corresponds in the range where the alpha varied between 1.0 and 0.7, the value of background-background probability $P(b|b)$ varied between 94.47% and 99.19%, a error less than 5.53%, considering that the foreground-foreground probability $P(f|f)$ was of 100.00%.

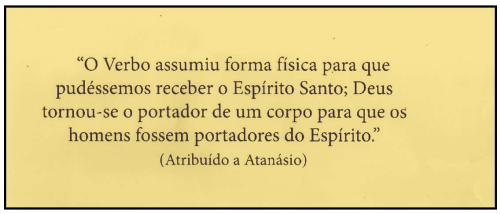
3.3.8 Silva-Lins-Rocha Approach

Silva (2006) considered the histogram distribution as the 256-symbol source (a priori source) distribution. Here it is further assumed the hypothesis, as in Pun (1981), that all symbols are statistically independent. In the case of real images one knows that this hypothesis does not hold. However, this largely simplifies the algorithm and yields good results.

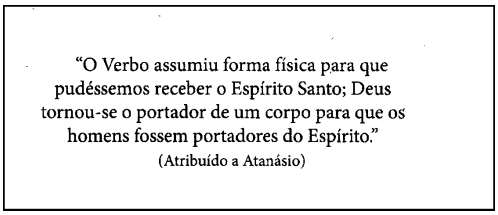
The result of applying Silva-Lins-Rocha Approach to the document image of Figure 3.15 is showed in Figure 3.25.



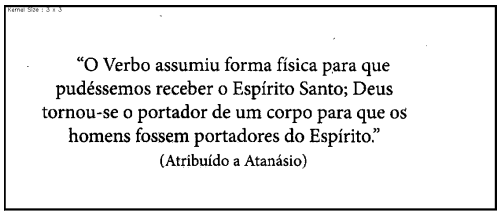
(a) Original Input Image



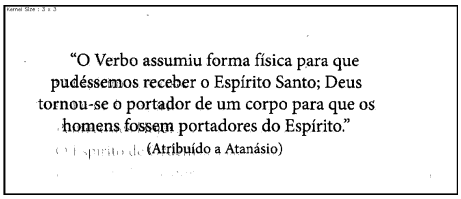
(b) Synthetic Image to be filtered



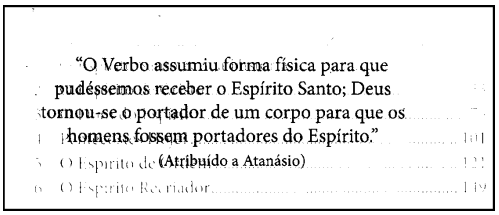
(c) Filtered Image with $\alpha = 1$



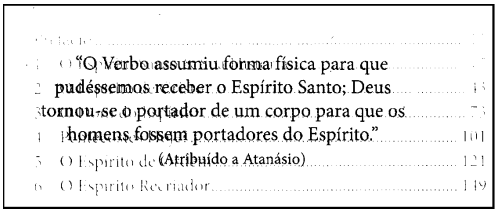
(d) Filtered Image with $\alpha = 0.9$



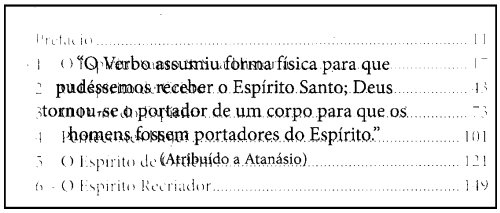
(e) Filtered Image with $\alpha = 0.8$



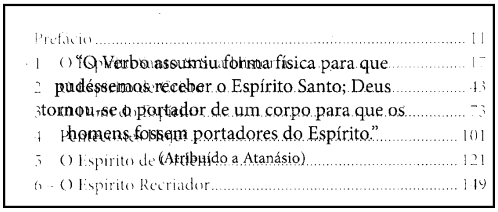
(f) Filtered Image with $\alpha = 0.7$



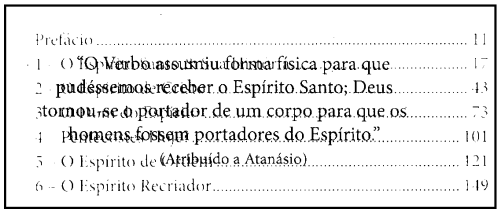
(g) Filtered Image with $\alpha = 0.6$



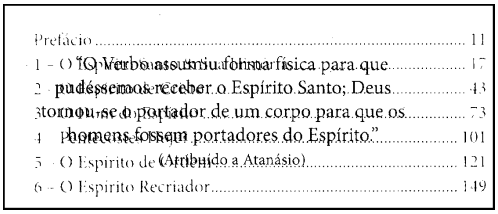
(h) Filtered Image with $\alpha = 0.5$



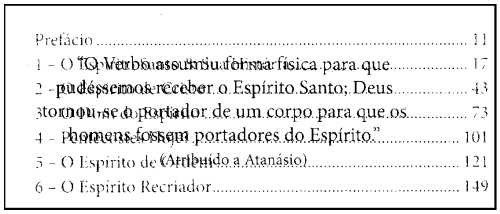
(i) Filtered Image with $\alpha = 0.4$



(j) Filtered Image with $\alpha = 0.3$



(k) Filtered Image with $\alpha = 0.2$



(l) Filtered Image with $\alpha = 0.1$

Figure 3.25: Silva-Lins-Rocha filtering of the images in Figure 3.15.

Table 3.9 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

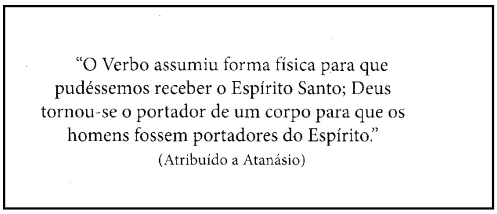
Table 3.9: Silva-Lins-Rocha Filter Result.

α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	89	97.60%	2.40%	78.73%	21.27%
0.2	1x1	153	233	252	3x3	95	97.77%	2.23%	82.80%	17.20%
0.3	1x1	153	233	252	3x3	105	97.94%	2.06%	86.73%	13.27%
0.4	1x1	153	233	252	3x3	115	98.17%	1.83%	90.60%	9.40%
0.5	1x1	153	233	252	3x3	126	98.44%	1.56%	94.96%	5.04%
0.6	1x1	153	233	252	3x3	137	98.80%	1.20%	99.22%	0.78%
0.7	1x1	153	233	252	3x3	150	98.80%	1.20%	100.00%	0.00%
0.8	1x1	153	233	252	3x3	161	98.98%	1.02%	100.00%	0.00%
0.9	1x1	153	233	252	3x3	167	99.07%	0.93%	100.00%	0.00%
1.0	1x1	153	233	252	3x3	165	99.26%	0.74%	100.00%	0.00%

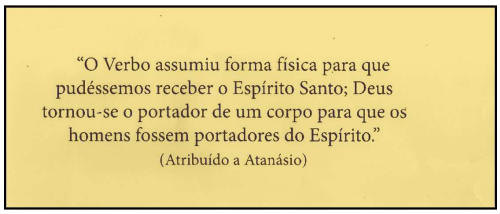
For visual analysis of the binarized images in Figure 3.25, it seems reasonable to consider important features such as partial elimination of back-to-front interference in Figure 3.25 (c) to (f), which corresponds, in the range, where the alpha varied between 1.0 to 0.7, the value of background-background probability $P(b|b)$ varied between 99.26% and 98.80%, respectively, a error less than 1.20%, considering that the foreground-foreground probability $P(f|f)$ was of 100.00%. While from the images in Figure 3.25 (g) to (l) the back-to-front interference is present, the proportion of an alpha reduction.

3.3.9 Wu-Lu Algorithm

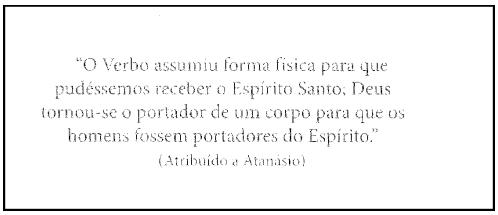
The result of applying Wu-Lu Algorithm, Wu (1998), to the document image of Figure 3.15 is showed in Figure 3.26.



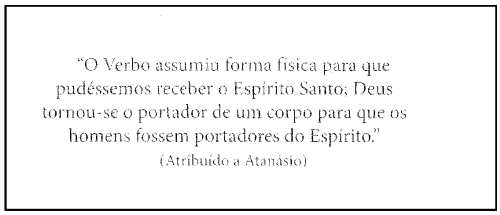
(a) Original Input Image



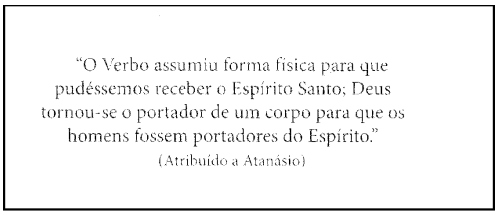
(b) Synthetic Image to be filtered



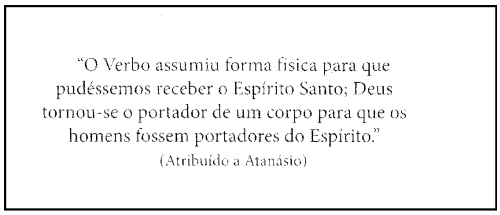
(c) Filtered Image with $\alpha = 1$



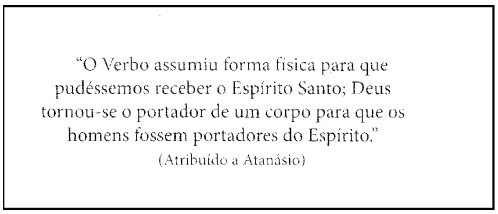
(d) Filtered Image with $\alpha = 0.9$



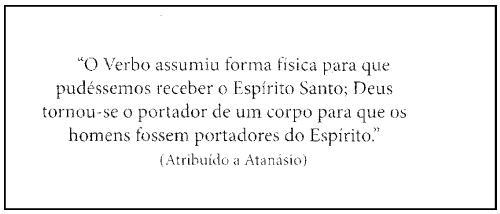
(e) Filtered Image with $\alpha = 0.8$



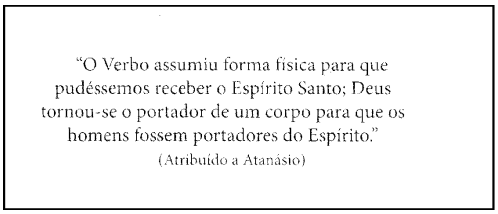
(f) Filtered Image with $\alpha = 0.7$



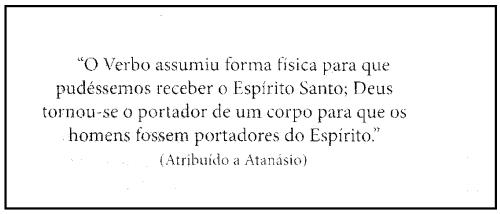
(g) Filtered Image with $\alpha = 0.6$



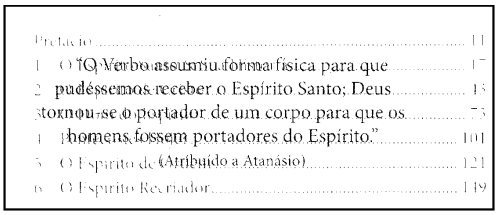
(h) Filtered Image with $\alpha = 0.5$



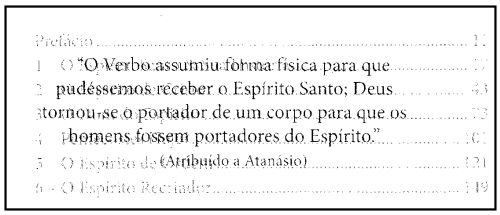
(i) Filtered Image with $\alpha = 0.4$



(j) Filtered Image with $\alpha = 0.3$



(k) Filtered Image with $\alpha = 0.2$



(l) Filtered Image with $\alpha = 0.1$

Figure 3.26: Wu-Lu filtering of the images in Figure 3.15.

Table 3.10 presents an analysis of the evolution of the different opacity coefficient α values versus the matrix of co-occurrence probability.

Table 3.10: Wu-Lu Filter Result.

α	kernel-Gaussian	Red	Green	Blue	kernel-Blur	Threshold	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	153	233	252	3x3	75	99.13%	0.87%	62.81%	37.19%
0.2	1x1	153	233	252	3x3	75	99.00%	1.00%	62.45%	37.55%
0.3	1x1	153	233	252	3x3	74	99.96%	0.04%	61.00%	39.00%
0.4	1x1	153	233	252	3x3	73	100.00%	0.00%	59.72%	40.28%
0.5	1x1	153	233	252	3x3	72	100.00%	0.00%	57.70%	42.30%
0.6	1x1	153	233	252	3x3	71	100.00%	0.00%	55.86%	44.14%
0.7	1x1	153	233	252	3x3	70	100.00%	0.00%	54.23%	45.77%
0.8	1x1	153	233	252	3x3	68	100.00%	0.00%	50.21%	49.79%
0.9	1x1	153	233	252	3x3	66	100.00%	0.00%	45.99%	54.01%
1.0	1x1	153	233	252	3x3	62	100.00%	0.00%	36.61%	63.39%

Although the value of background-background probability $P(b|b)$ does not vary and remains constant at 100.00%, where the alpha varied between 1.0 to 0.4, the value of foreground-foreground probability $P(f|f)$ varied between 36.61% and 59.72%, registering an error up to 63.39% in this range, displaying a loss of information in the text, when we do a visual analysis.

3.4 Classification of Binarized Images of the Filters Studied

The 16 documents shown in Figures 3.5 and 3.6, gave rise to 16 textures used to generate synthetic documents. Each of them were overlapped with another image to generate the document with 10 different intensities of α. The resulting 160 synthetic images were filtered using the following nine methods to remove back-to-front interference: Isodata, Pun, Kapur-Sahoo-Wong, Johannsen-Bille, Yen-Chang-Chang, Otsu, Mello-Lins, Silva-Lins-Rocha and Wu-Lu. At yielding 1,440 binarized images. To access the quality of the resulting images, the 1,440 matrices of co-occurrence probability were calculated.

Analyzing those $1,440$ matrices of co-occurrence probability, one can find the best filters for the different textures of historical documents with back-to-front interference. The loss condition of less than 1.00% error was adopted thus $P(f|f) \geq 99.00\%$ and $P(b|b) \geq 99.00\%$. For each historical document, one can select a filter with the optimal threshold. The results are shown in Table 3.11.

Table 3.11: Result of the Studied Filters with Synthetic Documents.

Red	Green	Blue	Grayscale	σ	Mode	σ-aging	alpha	p(b|b)	p(f|f)	Filter	Threshold
252	233	153	230	5	231	3	0.8	99.94	99.23	IsoData	137
252	233	153	230	5	231	3	0.9	99.98	99.20	IsoData	137
252	233	153	230	5	231	3	1.0	100.00	99.56	IsoData	138
252	233	153	230	5	231	3	0.7	99.25	100.00	Kapur-Sahoo-Wong	147
252	233	153	230	5	231	3	0.7	99.87	99.54	Otsu	138
252	233	153	230	5	231	3	0.8	99.94	99.56	Otsu	138
252	233	153	230	5	231	3	0.9	99.97	99.53	Otsu	138
252	233	153	230	5	231	3	1.0	99.95	99.56	Otsu	140
252	233	153	230	5	231	3	0.8	98.98	100.00	Silva-Lins-Rocha	161
252	233	153	230	5	231	3	0.9	99.07	100.00	Silva-Lins-Rocha	167
252	233	153	230	5	231	3	1.0	99.26	100.00	Silva-Lins-Rocha	165
252	233	153	230	5	231	3	0.9	99.19	100.00	Mello-Lins	165
253	238	158	233	3	233	3	0.7	99.75	99.45	Otsu	141
253	238	158	233	3	233	3	0.8	99.92	99.16	Otsu	140
253	238	158	233	3	233	3	0.9	99.97	99.14	Otsu	140
253	238	158	233	3	233	3	0.9	99.04	100.00	Silva-Lins-Rocha	171
253	238	158	233	3	233	3	0.9	99.29	100.00	Mello-Lins	165
241	200	136	205	4	206	5	0.7	99.19	99.53	IsoData	122
241	200	136	205	4	206	5	0.8	99.84	99.55	IsoData	120
241	200	136	205	4	206	5	0.9	99.87	99.55	IsoData	121
241	200	136	205	4	206	5	0.8	99.79	99.55	Otsu	122
241	200	136	205	4	206	5	0.9	99.81	99.55	Otsu	123
241	200	136	205	4	206	5	0.9	99.04	100.00	Silva-Lins-Rocha	147
245	204	141	209	4	211	5	0.7	99.59	99.60	IsoData	124
245	204	141	209	4	211	5	0.8	99.87	99.65	IsoData	124
245	204	141	209	4	211	5	0.9	99.92	99.64	IsoData	124
245	204	141	209	4	211	5	0.7	99.19	99.60	Otsu	127
245	204	141	209	4	211	5	0.8	99.84	99.65	Otsu	125
245	204	141	209	4	211	5	0.9	99.87	99.64	Otsu	126
245	204	141	209	4	211	5	0.9	99.04	100.00	Silva-Lins-Rocha	152
201	181	134	182	7	184	7	0.9	99.09	100.00	Silva-Lins-Rocha	127
254	247	208	244	4	246	5	0.7	99.48	100.00	Kapur-Sahoo-Wong	164
254	247	208	244	4	246	5	0.8	99.17	100.00	Kapur-Sahoo-Wong	180
254	247	208	244	4	246	5	0.9	99.04	100.00	Silva-Lins-Rocha	126
254	247	208	244	4	246	5	1.0	99.27	100.00	Silva-Lins-Rocha	123
254	247	208	244	4	246	5	0.8	99.67	100.00	Mello-Lins	168
254	247	208	244	4	246	5	0.9	99.84	100.00	Mello-Lins	164
254	247	208	244	4	246	5	1.0	99.45	100.00	Mello-Lins	179
253	232	152	229	4	229	3	0.7	99.78	99.21	IsoData	136
253	232	152	229	4	229	3	0.8	99.92	99.21	IsoData	136
253	232	152	229	4	229	3	0.9	99.97	99.20	IsoData	136
253	232	152	229	4	229	3	1.0	100.00	99.51	IsoData	137
253	232	152	229	4	229	3	0.8	99.07	100.00	Kapur-Sahoo-Wong	157
253	232	152	229	4	229	3	0.7	99.59	99.53	Otsu	139
253	232	152	229	4	229	3	0.8	99.87	99.54	Otsu	139
253	232	152	229	4	229	3	0.9	99.92	99.52	Otsu	139
253	232	152	229	4	229	3	1.0	99.95	99.51	Otsu	139
253	232	152	229	4	229	3	0.9	99.04	100.00	Silva-Lins-Rocha	167
253	232	152	229	4	229	3	1.0	99.27	100.00	Silva-Lins-Rocha	164
253	232	152	229	4	229	3	0.9	99.14	100.00	Mello-Lins	165
253	223	156	224	6	227	5	1.0	100.00	99.09	IsoData	135
253	223	156	224	6	227	5	0.8	99.07	100.00	Kapur-Sahoo-Wong	156
253	223	156	224	6	227	5	0.7	99.75	99.45	Otsu	136
253	223	156	224	6	227	5	0.8	99.92	99.14	Otsu	136
253	223	156	224	6	227	5	0.9	99.97	99.46	Otsu	136
253	223	156	224	6	227	5	1.0	99.95	99.48	Otsu	138
253	223	156	224	6	227	5	0.9	99.04	100.00	Silva-Lins-Rocha	166
253	223	156	224	6	227	5	1.0	99.27	100.00	Silva-Lins-Rocha	163
253	223	156	224	6	227	5	0.9	99.09	100.00	Mello-Lins	165
248	197	132	205	6	207	7	1.0	100.00	99.09	IsoData	124
248	197	132	205	6	207	7	0.8	99.07	100.00	Kapur-Sahoo-Wong	140
248	197	132	205	6	207	7	0.7	99.75	99.45	Otsu	132
248	197	132	205	6	207	7	0.8	99.92	99.14	Otsu	125
248	197	132	205	6	207	7	0.9	99.97	99.46	Otsu	126
248	197	132	205	6	207	7	1.0	99.95	99.48	Otsu	126
248	197	132	205	6	207	7	0.9	99.04	100.00	Silva-Lins-Rocha	149
248	197	132	205	6	207	7	1.0	99.27	100.00	Silva-Lins-Rocha	146
248	197	132	205	6	207	7	0.9	99.09	100.00	Mello-Lins	165
212	192	141	192	7	193	7	0.7	99.34	99.55	IsoData	121
212	192	141	192	7	193	7	1.0	99.91	99.58	IsoData	112
212	192	141	192	7	193	7	0.8	99.79	99.56	Otsu	113
212	192	141	192	7	193	7	0.9	99.84	99.58	Otsu	113
212	192	141	192	7	193	7	1.0	99.88	99.58	Otsu	113
212	192	141	192	7	193	7	0.9	99.09	100.00	Silva-Lins-Rocha	137
212	192	141	192	7	193	7	1.0	99.30	100.00	Silva-Lins-Rocha	134
242	243	247	243	5	246	5	0.7	99.59	100.00	Kapur-Sahoo-Wong	171
242	243	247	243	5	246	5	0.8	99.17	100.00	Kapur-Sahoo-Wong	188
242	243	247	243	5	246	5	0.9	99.04	100.00	Silva-Lins-Rocha	199
242	243	247	243	5	246	5	1.0	99.27	100.00	Silva-Lins-Rocha	196
242	243	247	243	5	246	5	0.7	99.34	100.00	Mello-Lins	173
242	243	247	243	5	246	5	0.8	99.89	100.00	Mello-Lins	170
242	243	247	243	5	246	5	1.0	99.67	100.00	Mello-Lins	182
226	219	207	220	4	219	3	0.7	99.48	100.00	Kapur-Sahoo-Wong	145
226	219	207	220	4	219	3	0.8	99.17	100.00	Kapur-Sahoo-Wong	161
226	219	207	220	4	219	3	0.9	99.04	100.00	Silva-Lins-Rocha	172
226	219	207	220	4	219	3	1.0	99.27	100.00	Silva-Lins-Rocha	169
226	219	207	220	4	219	3	0.9	99.29	100.00	Mello-Lins	166

3.5 Conclusions

The 16 historical documents from the bequest of Joaquim Nabuco bequest and the SBrT file were used to generate 160 synthesised images with different textures and 10 different strengths of back-to-front interference ($0.1 \leq \alpha \leq 1.0$) such images were passed through the nine filters studied: Isodata, Pun, Kapur-Sahoo-Wong, Johannsen-Bille, Yen-Chang-Chang, Otsu, Mello-Lins, Silva-Lins-Rocha and Wu-Lu, generating a database of 1,440 binarized documents.

Of the 16 historical documents analyzed, 12 could have some treatment by the studied filters, under the imposed conditions of the co-occurrence probabilities $P(b|b) \geq 99.00\%$ and P $(f|f) \geq 99.00\%$ shown as an example in Section 3.4. In each binarized document one has one or more filters selected with the optimum threshold and corresponding co-occurrence probabilities $P(b|b)$ and $(f|f)$, providing a measure of the quality of the binarized images. The remaining 4 historical documents the filters were not able to meet the above conditions.

The Pun method was not efficient in filtering the 160 images generated synthetically, because the co-occurrence probability $P(b|f)$ or maximum error was above 32.00%. The visual inspection of the binarized documents also corroborate to this.

Thus, from the visual aspect of the binarized images studied as well as the matrix co-occurrence probability, one finds that the remaining algorithms are good thresholding methods to remove back-to-from interference in historical documents, depending on the texture of the historical document and the intensity of α of the back-to-front interference, according to the results shown in Table 3.11.

The filters that met the requirements of co-occurrence probabilities in the given example were: Isodata, Otsu, Kapur-Sahoo-Wong, Silva-Lins-Rocha and Mello-Lins. The opacity factor value α varied from 0.7 to 1.0.

4 A NEW BINARIZATION ALGORITHM FOR IMAGES WITH BACK-TO-FRONT INTERFERENCE

The previous Chapter presented the problem of back-to-front interference and assessed some of the different techniques to filter out such a noise in document images. This Chapter presents a new algorithm to remove such a noise and assesses its applicability.

4.1 The New Algorithm

The technical literature presents for document image processing several thresholding algorithms, both with and without back-to-front interference. None of them has been shown to be effective for all strengthes of interference, textures of the background, etc, (Leedham, 2002). This Chapter presents a new algorithm for filtering out the back-to-front interference, which examines both pixels of the foreground and the background regions.

The proposed filter adopted a global approach using four steps, as shown the block diagram in Figure 4.1.

1. Filtering using the Bilateral filter.
2. Split image in RGB components.
3. Decision-making block for each RGB channel.
4. Classify the binarized images.

The present study investigates the impact of selecting of proposed filter variables in order to remove the noise and back-to-front interference in the historical documents. A bilateral filter (whose details are described in a Appendix A) was used due its property of de-noising digital images while preserving the edges of the written/typed areas of the document. The RGB components of the image were used to analyze which of them best preserved the text information in the foreground. An adaptive binarization method inspired by Otsu's method was implemented, with a minute choice of thresholding level,

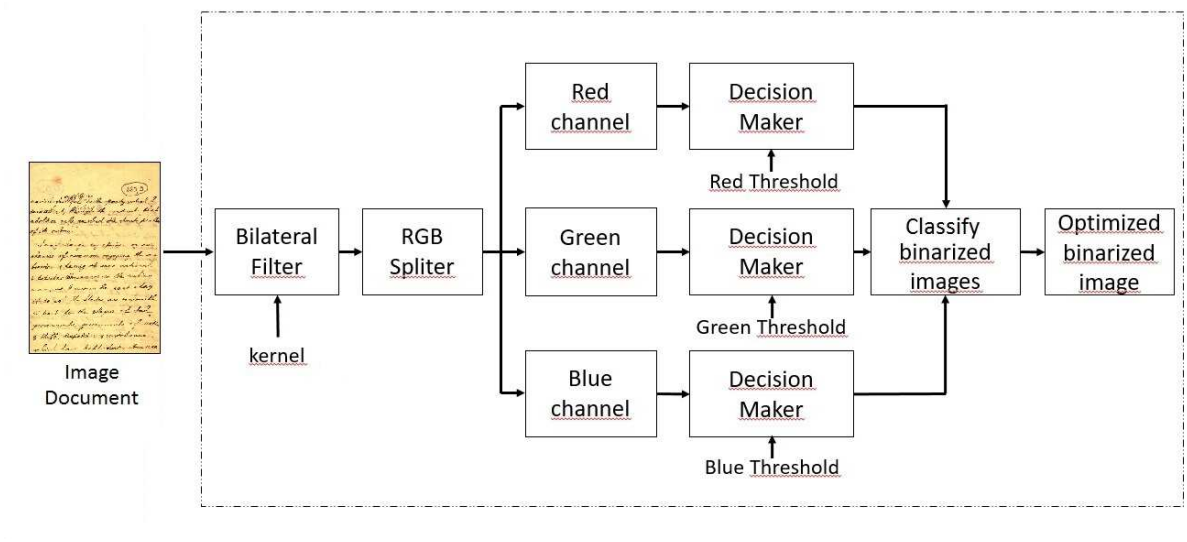


Figure 4.1: Block Diagram of the proposed method.

as a priori information to differentiate between text and non-text regions, and the last step was used to analyze and to decide which of the RGB components best preserved the text information in the foreground.

4.2 The Decision Making Block

After passing through the bilateral filter, the image is then divided by the splitter into its Red, Green and Blue components, as shown the block diagram in Figure 4.1. The preservation of image information is a feature of this stage, because the image filtered by the bilateral filter is still in color, unlike the transformation process grayscale, which introduces a quantization error, with losses of image information due to metamerism. Once the RGB channels are generated, the decision-making block is applied to process and the optimal threshold is calculated for each RGB channel, then three binary images are generated. The background-background probability is a function of RGB texture as well as the alpha opacity factor. Thus, we can represent this dependence as shown in the Equation 4.2.1:

$$P(b, b) = f(\alpha, R, G, B) \tag{4.2.1}$$

The optimal threshold t^* for each channel is calculated in the decision making block, as shown in the Equation 4.2.2:

$$t^* = \text{Max}\, P(b|b) \tag{4.2.2}$$

subject to

$$P(f|f) \geq 98\%$$

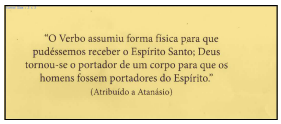
The criterion used was $P(f|f) \geq 98.00\%$. That is, only 2% of undesirable pixels are allowed in the background, while we will search for the maximum of black pixels in the text. The matrix of co-occurrence probability are calculated and the decision maker chooses the best binarized image.

4.3 Results of the Proposed Method

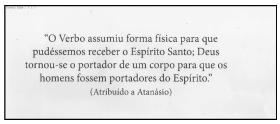
Figure 3.15 in Chapter 3 shows the document synthesized with back-to-front interference and aging for various α opacity coefficient values. The master original image has $521,920$ pixels of which $20,529$ are black and $501,391$ are white. Thus, the probability distribution is P(white pixels)= 96.07% and P(black pixels)= 3.93%.

The proposed method was applied to the document image in Figure 3.15 following the block diagram in Figure 4.1.

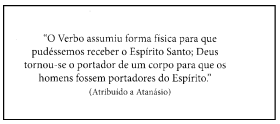
Were ran several sets of experiments on synthetically generated images sets to analyze the performance of the method here. Figure 4.2 show the results of this evaluation for the Red channel.



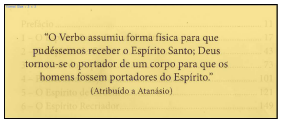
(a) Synthetic Image with $\alpha = 1$



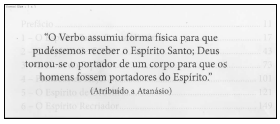
(b) Output Image Red channel with $\alpha = 1$



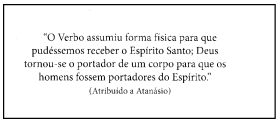
(c) Binarized Image with $\alpha = 1$



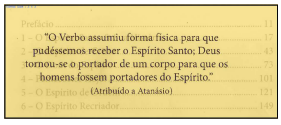
(d) Synthetic Image with $\alpha = 0.9$



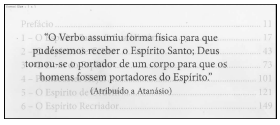
(e) Output Image Red channel with $\alpha = 0.9$



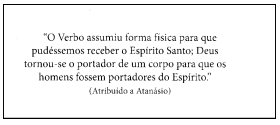
(f) Binarized Image with $\alpha = 0.9$



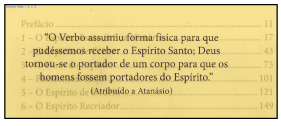
(g) Synthetic Image with $\alpha = 0.8$



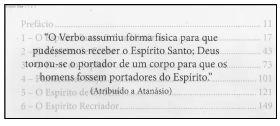
(h) Output Image Red channel with $\alpha = 0.8$



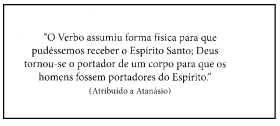
(i) Binarized Image with $\alpha = 0.8$



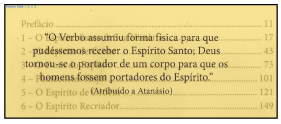
(j) Synthetic Image with $\alpha = 0.7$



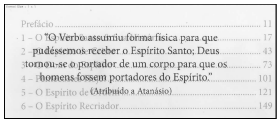
(k) Output Image Red channel with $\alpha = 0.7$



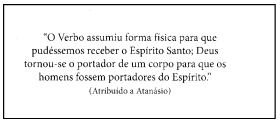
(l) Binarized Image with $\alpha = 0.7$



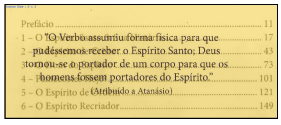
(m) Synthetic Image with $\alpha = 0.6$



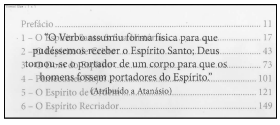
(n) Output Image Red channel with $\alpha = 0.6$



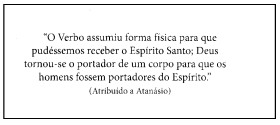
(o) Binarized Image with $\alpha = 0.6$



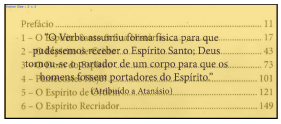
(p) Synthetic Image with $\alpha = 0.5$



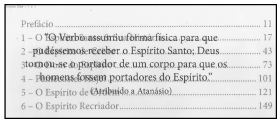
(q) Output Image Red channel with $\alpha = 0.5$



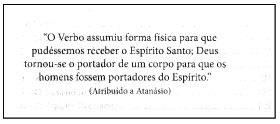
(r) Binarized Image with $\alpha = 0.5$



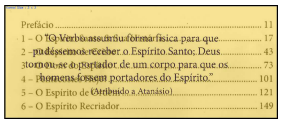
(s) Synthetic Image with $\alpha = 0.4$



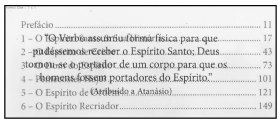
(t) Output Image Red channel with $\alpha = 0.4$



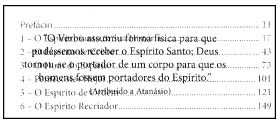
(u) Binarized Image with $\alpha = 0.4$



(v) Synthetic Image with $\alpha = 0.3$



(w) Output Image Red channel with $\alpha = 0.3$



(x) Binarized Image with $\alpha = 0.3$

Figure 4.2: Input and Output images of the Red channel using the proposed filter.

For visual analysis of the binarized images in Figure 4.2 (c), (f), (i), (l), (o), (r) and (u), it seems reasonable to consider important features for removing back-to-front interference, which corresponds, in the range, where the alpha varied between 1.0 to 0.4. While from the images in Figure 4.2 (x) the back-to-front interference is present, the proportion of an alpha reduction.

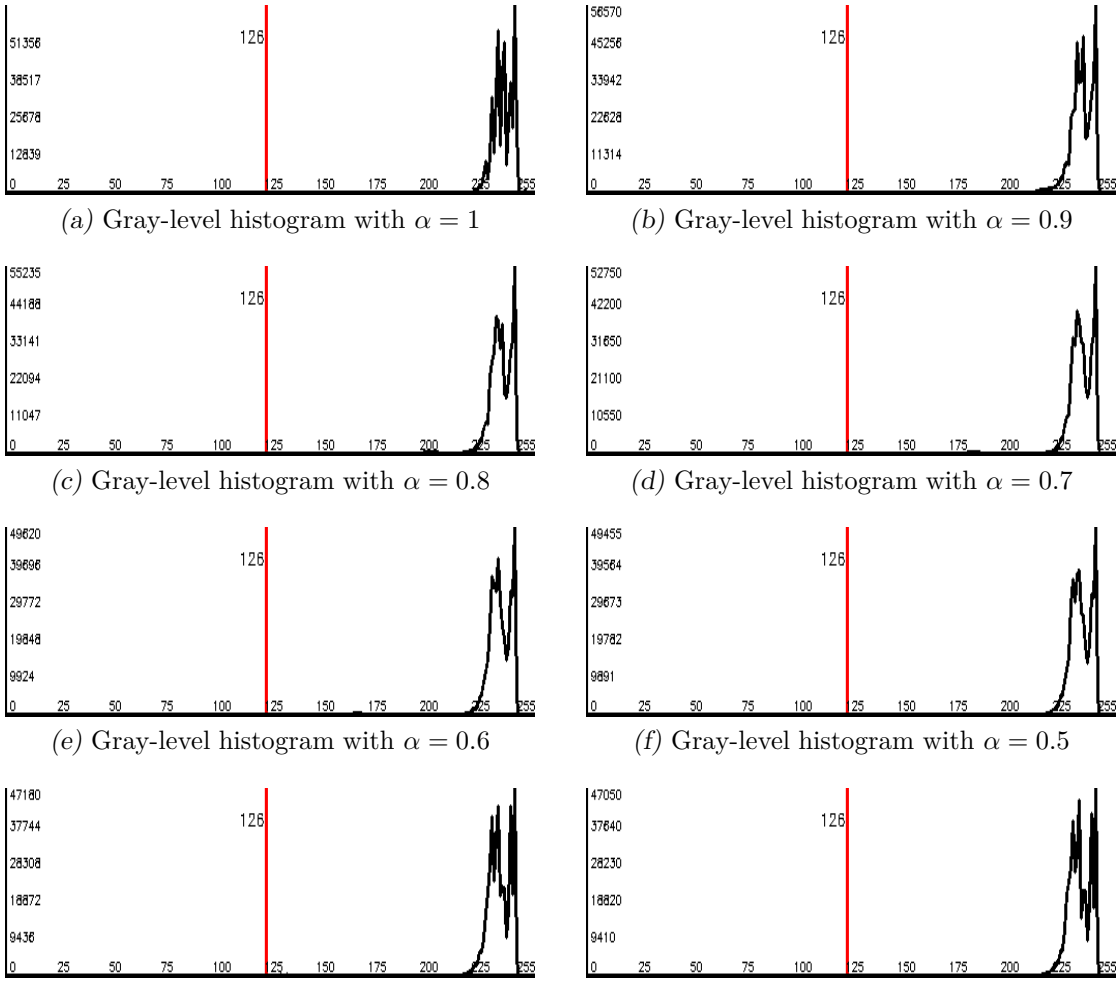
Table 4.1 presents an analysis of the variation of the opacity coefficient α versus the co-occurrence probability matrix for the Red channel.

Table 4.1: Output proposed filter for the Red channel.

α	kernel-Gauss	Red	Green	Blue	kernel-Blur	threshold	kernel-Bil	P(b|b)	P(b|f)	P(f|f)	P(f|b)
0.1	1x1	154	233	253	3x3	126	1x1	96.49%	3.51%	100.00%	0.00%
0.2	1x1	154	233	253	3x3	126	1x1	96.93%	3.07%	100.00%	0.00%
0.3	1x1	154	233	253	3x3	126	1x1	97.66%	2.34%	100.00%	0.00%
0.4	1x1	154	233	253	3x3	126	1x1	99.60%	0.40%	100.00%	0.00%
0.5	1x1	154	233	253	3x3	126	1x1	99.87%	0.13%	100.00%	0.00%
0.6	1x1	154	233	253	3x3	126	1x1	99.91%	0.09%	100.00%	0.00%
0.7	1x1	154	233	253	3x3	126	1x1	99.94%	0.06%	100.00%	0.00%
0.8	1x1	154	233	253	3x3	126	1x1	99.97%	0.03%	100.00%	0.00%
0.9	1x1	154	233	253	3x3	126	1x1	99.99%	0.01%	100.00%	0,00%
1.0	1x1	154	233	253	3x3	126	1x1	100.00%	0.00%	100.00%	0.00%

Analyzing the co-occurrence probability matrix of the data produced by the proposed filter in Table 4.1, where the alpha varied between 1.0 and 0.4, the value of probability of background-background $P(b|b)$ varied between 100.00% and 99.60%, respectively, a error less than 0.40%, considering that the probability of co-occurrence $P(f|f)$ remained constant at 100%. Such results mean that the back-to-front interference was almost completely removed and the textual information was almost fully preserved. The results are consistent with the visual inspection of the images shown in Figure 4.2.

Figure 4.3 shows the decision-making block response for the optimal threshold selection t^*. For the Red channel the value that maximizes the background-background probability $P(b|b)$ was chosen, whose value was $t^*_{Red} = 126$, as shown by the grayscale histogram in Figure 4.3.



(a) Gray-level histogram with $\alpha = 1$

(b) Gray-level histogram with $\alpha = 0.9$

(c) Gray-level histogram with $\alpha = 0.8$

(d) Gray-level histogram with $\alpha = 0.7$

(e) Gray-level histogram with $\alpha = 0.6$

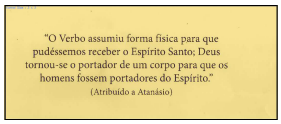
(f) Gray-level histogram with $\alpha = 0.5$

(g) Gray-level histogram with $\alpha = 0.4$

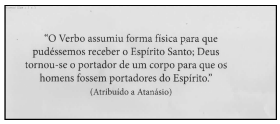
(h) Gray-level histogram with $\alpha = 0.3$

Figure 4.3: Gray-level Images Histogram Red channel from proposed filter.

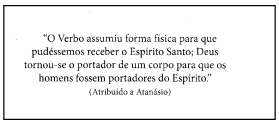
Figure 4.4 shows the results of the variation of the α for the Green channel.



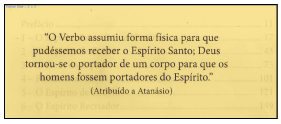
(a) Synthetic Image with $\alpha = 1$



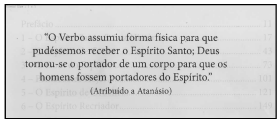
(b) Output Image Green channel with $\alpha = 1$



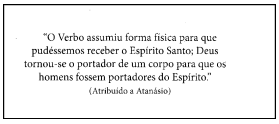
(c) Binarized Image with $\alpha = 1$



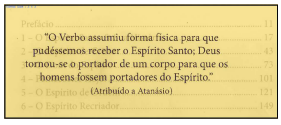
(d) Synthetic Image with $\alpha = 0.9$



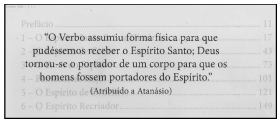
(e) Output Image Green channel with $\alpha = 0.9$



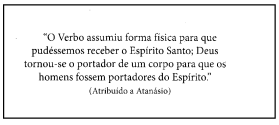
(f) Binarized Image with $\alpha = 0.9$



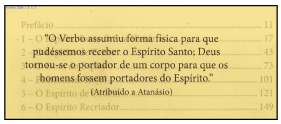
(g) Synthetic Image with $\alpha = 0.8$



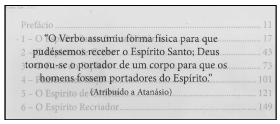
(h) Output Image Green channel with $\alpha = 0.8$



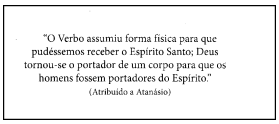
(i) Binarized Image with $\alpha = 0.8$



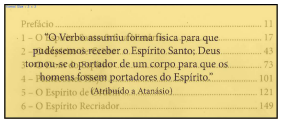
(j) Synthetic Image with $\alpha = 0.7$



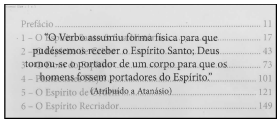
(k) Output Image Green channel with $\alpha = 0.7$



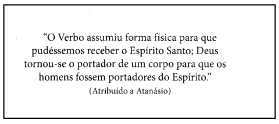
(l) Binarized Image with $\alpha = 0.7$



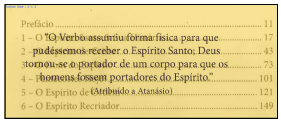
(m) Synthetic Image with $\alpha = 0.6$



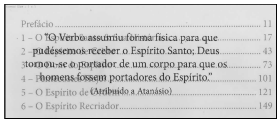
(n) Output Image Green channel with $\alpha = 0.6$



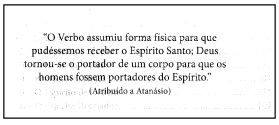
(o) Binarized Image with $\alpha = 0.6$



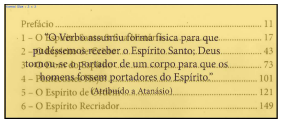
(p) Synthetic Image with $\alpha = 0.5$



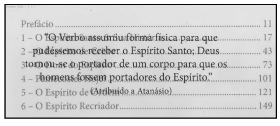
(q) Output Image Green channel with $\alpha = 0.5$



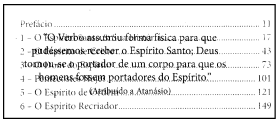
(r) Binarized Image with $\alpha = 0.5$



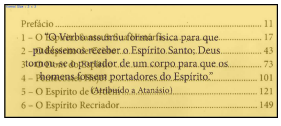
(s) Synthetic Image with $\alpha = 0.4$



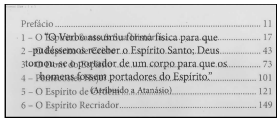
(t) Output Image Green channel with $\alpha = 0.4$



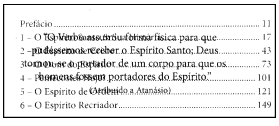
(u) Binarized Image with $\alpha = 0.4$



(v) Synthetic Image with $\alpha = 0.3$



(w) Output Image Green channel with $\alpha = 0.3$



(x) Binarized Image with $\alpha = 0.3$

Figure 4.4: Input and Output images of the Green channel using the proposed filter.

The examination of the images in Figure 4.4 (c), (f), (i), (l), (o) and (r), allows to observe the removal of the back-to-front interference, which corresponds, to the range of variation of the α between 1.0 and 0.5. The images in Figure 4.4 (u) and (x) shows that the back-to-front interference is present.

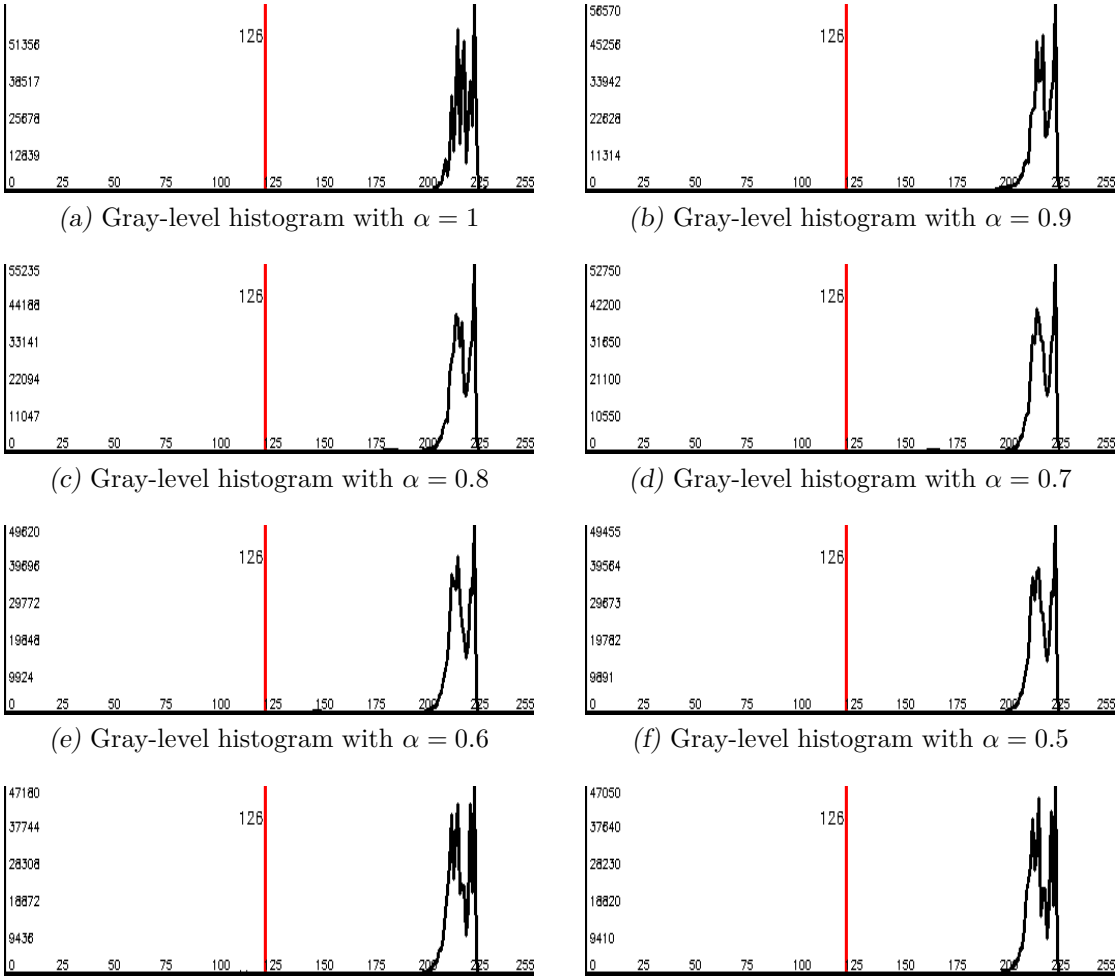
Table 4.2 presents an analysis of the variation of the opacity coefficient α versus the matrix of co-occurrence probability for Green channel.

Table 4.2: Output of the proposed filter for the Green channel.

α	kernel-Gauss	Red	Green	Blue	kernel-Blur	threshold	kernel-Bil	P(bb)	P(bf)	P(ff)	P(fb)
0.1	1x1	154	233	253	3x3	126	1x1	95.52%	4.48%	100.00%	0.00%
0.2	1x1	154	233	253	3x3	126	1x1	95.87%	4.13%	100.00%	0.00%
0.3	1x1	154	233	253	3x3	126	1x1	96.32%	3.68%	100.00%	0.00%
0.4	1x1	154	233	253	3x3	126	1x1	97.00%	3.00%	100.00%	0.00%
0.5	1x1	154	233	253	3x3	126	1x1	99.10%	0.90%	100.00%	0.00%
0.6	1x1	154	233	253	3x3	126	1x1	99.44%	0.56%	100.00%	0.00%
0.7	1x1	154	233	253	3x3	126	1x1	99.49%	0.51%	100.00%	0.00%
0.8	1x1	154	233	253	3x3	126	1x1	99.52%	0.48%	100.00%	0.00%
0.9	1x1	154	233	253	3x3	126	1x1	99.54%	0.46%	100.00%	0.00%
1.0	1x1	154	233	253	3x3	126	1x1	99.57%	0.43%	100.00%	0.00%

The co-occurrence probability matrix of the data produced by the proposed filter is shown in Table 4.2, in which the value of the α varied between 1.0 and 0.5, the value of the probability of background-background $P(b|b)$ varied between 100.00% and 99.10%, respectively, a error of less than 0.90%, considering that the co-occurrence probability $P(f|f)$ remained constant at 100%, meaning that back-to-front interference was almost completely removed and that the textual information was fully preserved, in this range. Such results are consistent with the visual inspection of images in Figure 4.4.

Figure 4.5 shows the decision-making block response for the optimal threshold selection t^*. For the Green channel the value that maximizes the background-background probability $P(b|b)$, was $t^*_{Green} = 126$, as shown by the grayscale histogram in Figure 4.5.



(a) Gray-level histogram with $\alpha = 1$

(b) Gray-level histogram with $\alpha = 0.9$

(c) Gray-level histogram with $\alpha = 0.8$

(d) Gray-level histogram with $\alpha = 0.7$

(e) Gray-level histogram with $\alpha = 0.6$

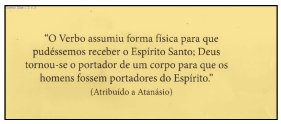
(f) Gray-level histogram with $\alpha = 0.5$

(g) Gray-level histogram with $\alpha = 0.4$

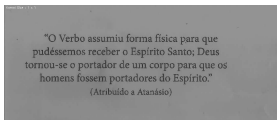
(h) Gray-level histogram with $\alpha = 0.3$

Figure 4.5: Output Images Histogram Green channel from proposed filter.

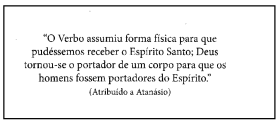
The output of the binarized images of the Blue channel is shown in Figure 4.6.



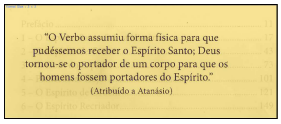
(a) Synthetic Image with $\alpha = 1$



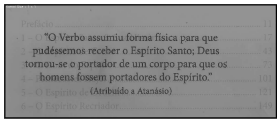
(b) Output Image Blue channel with $\alpha = 1$



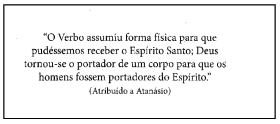
(c) Binarized Image with $\alpha = 1$



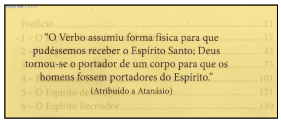
(d) Synthetic Image with $\alpha = 0.9$



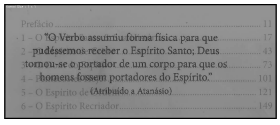
(e) Output Image Blue channel with $\alpha = 0.9$



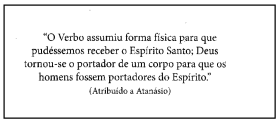
(f) Binarized Image with $\alpha = 0.9$



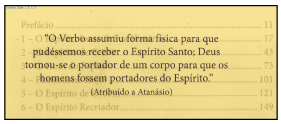
(g) Synthetic Image with $\alpha = 0.8$



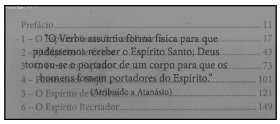
(h) Output Image Blue channel with $\alpha = 0.8$



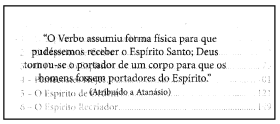
(i) Binarized Image with $\alpha = 0.8$



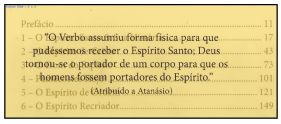
(j) Synthetic Image with $\alpha = 0.7$



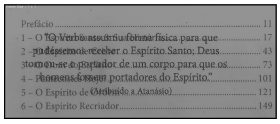
(k) Output Image Blue channel with $\alpha = 0.7$



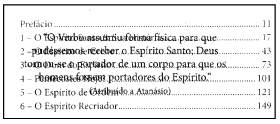
(l) Binarized Image with $\alpha = 0.7$



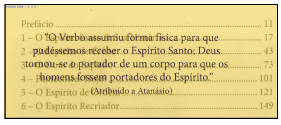
(m) Synthetic Image with $\alpha = 0.6$



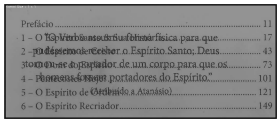
(n) Output Image Blue channel with $\alpha = 0.6$



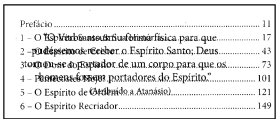
(o) Binarized Image with $\alpha = 0.6$



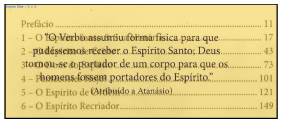
(p) Synthetic Image with $\alpha = 0.5$



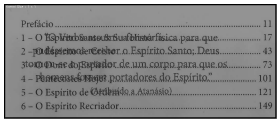
(q) Output Image Blue channel with $\alpha = 0.5$



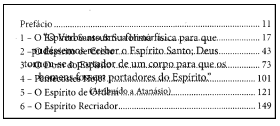
(r) Binarized Image with $\alpha = 0.5$



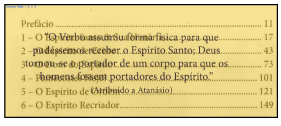
(s) Synthetic Image with $\alpha = 0.4$



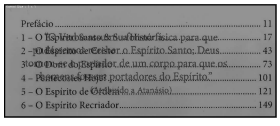
(t) Output Image Blue channel with $\alpha = 0.4$



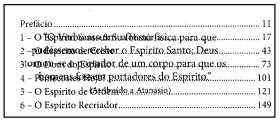
(u) Binarized Image with $\alpha = 0.4$



(v) Synthetic Image with $\alpha = 0.3$



(w) Output Image Blue channel with $\alpha = 0.3$



(x) Binarized Image with $\alpha = 0.3$

Figure 4.6: Input and Output images of the Blue channel using the proposed filter.

A closer look at the binarized images in Figure 4.6, allows to observe that the images of the Blue channel did not produce good quality binary images. Therefore, the visual inspection of such images do not recommend the use of the Blue components in removing the front-to back interference in documents.

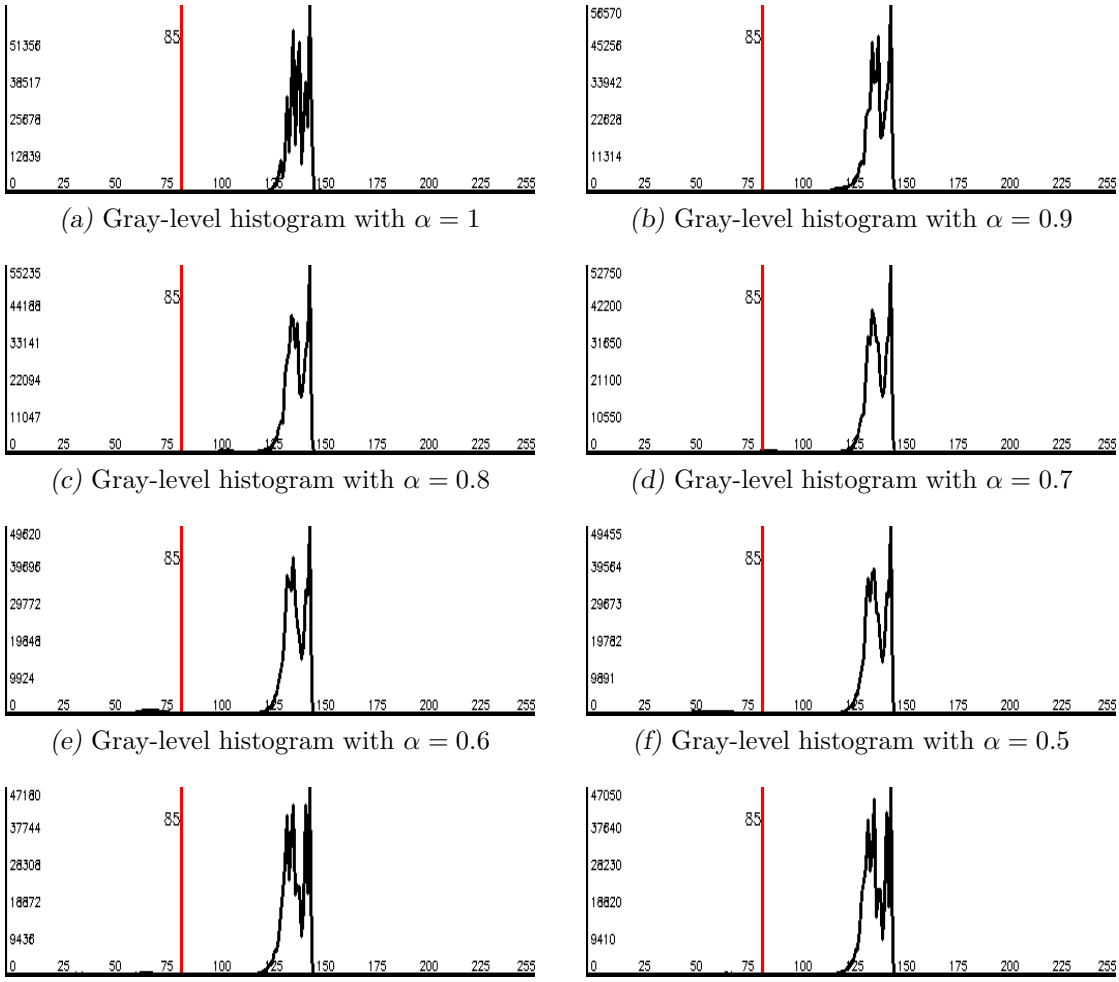
Table 4.3 presents an analysis of the variation of the opacity coefficient α versus the matrix of co-occurrence probability for the Blue channel.

Table 4.3: Output proposed filter for the Blue channel.

α	kernel-Gauss	Red	Green	Blue	kernel-Blur	threshold	kernel-Bil	P(bb)	P(bf)	P(ff)	P(fb)
0.1	1x1	154	233	254	3x3	85	1x1	96.87%	5.13%	90.32%	9.68%
0.2	1x1	154	233	254	3x3	85	1x1	94.76%	5.24%	89.59%	10.41%
0.3	1x1	154	233	254	3x3	85	1x1	94.10%	5.90%	87.05%	12.95%
0.4	1x1	154	233	254	3x3	85	1x1	94.39%	5.61%	86.68%	13.32%
0.5	1x1	154	233	254	3x3	85	1x1	94.83%	5.17%	86.42%	13.58%
0.6	1x1	154	233	254	3x3	85	1x1	95.53%	4.47%	86.32%	13.68%
0.7	1x1	154	233	254	3x3	85	1x1	97.64%	2.36%	86.42%	13.58%
0.8	1x1	154	233	254	3x3	85	1x1	98.56%	1.44%	86.45%	13.55%
0.9	1x1	154	233	254	3x3	85	1x1	98.62%	1.38%	86.47%	13.53%
1.0	1x1	154	233	254	3x3	85	1x1	98.66%	1.34%	86.51%	12.49%

The minimum value of the background-background probability P(b|b) presented in Table 4.3 was 94.10%, considering that the foreground-foreground probability P(f|f) varied between 90.32% and 86.32%, generating a loss of text information. Thus, the presented results show that the images that correspond to the Blue channel are not recommended for removing the front-to back interference, for any value of α.

Figure 4.7 shows the decision-making block response for the optimal threshold selection t^*. For the Blue channel the value that maximizes the background-background probability $P(b|b)$ was $t^*_{Blue} = 85$, as shown by the grayscale histogram in Figure 4.7.



(a) Gray-level histogram with $\alpha = 1$

(b) Gray-level histogram with $\alpha = 0.9$

(c) Gray-level histogram with $\alpha = 0.8$

(d) Gray-level histogram with $\alpha = 0.7$

(e) Gray-level histogram with $\alpha = 0.6$

(f) Gray-level histogram with $\alpha = 0.5$

(g) Gray-level histogram with $\alpha = 0.4$

(h) Gray-level histogram with $\alpha = 0.3$

Figure 4.7: Output Images Histogram for the Blue channel using the proposed filter.

4.4 Threshold Calculated using Regression Model

Based on the characteristics of the several historical documents, samples can provide enough statistical data to model the automatic calculation of the optimal threshold, using a regression analysis. From the set of data simulating with 160 documents synthesized it generated a threshold estimation model of the proposed method.

A sample of the historical document shown in Section 3.2 was used to measure the mean and variance values of the RGB texture measured using ImageJ. Table 4.4 shows

the results of this evaluation, the value of the mean, the standard deviation, the mode and the grayscale intensity of the RGB components, all such information is needed to calculate the filtering threshold.

Table 4.4: Texture information of historical document.

Label	Mean	Standard Deviation	Mode	Min	Max
Red	252.740	2.506	255	226	255
Green	233.209	6.252	234	193	248
Blue	153.654	4.907	155	126	167
GrayScale Intensity	229.99	4.65	231	196	241

The GrayScale intensity is calculated using in Equation 7.1.11

$$GrayScale - intensity = 0.299R + 0.587G + 0.114B \tag{4.4.1}$$

For the estimated value of the threshold $\hat{t}$ is given by the regression in Equation 7.1.12:

$$\begin{aligned} \hat{t} \;=\; & 1.80 + 8.57(\text{Red}) + 15.08(\text{Green}) + 3.00(\text{Blue}) - 25.44(\text{GrayScale-Intensity}) \\ & +7.78(\text{GrayScale-Standard-Deviation}) + 0.78(\text{Mode}) \\ & -6.73(\text{kernell aging}) \end{aligned} \tag{4.4.2}$$

The analysis of Tables 4.5 and 4.6 guarantee the choice of the predictors and also explain the prediction model. Analyzing Table 4.5 we find that the p-value of all predictors

Table 4.5: Analysis of Variance of Threshold.

Source Regression	DF	f-value	p-value
Predictors	7	56.19	0.000
kernell aging	1	31.08	0.000
Red	1	82.60	0.000
Green	1	84.58	0.000
Blue	1	69.16	0.000
Intensity mean	1	74.37	0.000
Intensity standard deviation	1	46.55	0.000
Mode	1	1.40	0.239

are less than ≤ 0.05, except for the predicted Mode. If the p-value is less than or equal to

Table 4.6: Model Summary.

S	R^2	R^2 (adjusted)	R^2 (predict)
8.53	72.13%	70.84%	69.59%

α, one may conclude that the effect is statistically significant. The R^2 value describes the amount of variation in the observed response values that is explained by the predictors in Table 4.6. The R^2 (predict) value indicates that the model explains 72.13% of the variation in threshold when we use it for prediction.

The 160 synthesized documents with several aging textures in addition to the alpha values ranging from 0.1 to 1.0 in steps of 0.1, were used for generating a database of images whose sample is shown in Table 4.7. For each synthesized historical document,

Table 4.7: Result of the Simulation with Synthetic Documents using the Proposed Filter.

Red	**Green**	**Blue**	**kernell (aging)**	**Grayscale**	**Standard**	**Mode**	**Threshold**	$\hat{t}$	**Error**
252	233	153	5	230	5	231	126	121,41	3,64%
180	165	123	11	165	13	165	105	102,41	2,47%
253	238	158	3	233	3	233	130	126,94	2,35%
241	200	136	5	205	4	206	116	112,76	2,79%
245	204	141	5	209	4	211	120	116,7	2,75%
201	181	134	7	182	7	184	78	89,6	-14,87%
224	174	113	5	182	4	182	98	109,83	-12,07%
254	247	208	5	244	4	246	121	125,57	-3,78%
253	232	152	3	229	4	229	127	131,12	-3,24%
154	128	89	7	131	7	132	100	90,57	9,43%
218	167	113	5	177	6	180	92	97,17	-5,62%
253	223	156	5	224	6	227	127	138,26	-8,87%
248	197	132	7	205	6	207	124	116,83	5,78%
212	192	141	7	192	7	193	87	109,33	-25,67%
242	243	247	5	243	5	246	116	112,63	2,91%
226	219	207	3	220	4	219	100	105,45	-5,45%

the prediction of threshold $\hat{t}$ was calculated. The "Error" column represents the relative error. The optimal threshold is calculated between the best threshold of R, G and B components.

The average Error, considering the training database of Table 4.7 is 6.98%.

4.5 Classification of Binarized Images Including the Proposed Filter

From these 16 documents shown in Figures 3.5 and 3.6, 160 images were generated based on synthetic documents similar to historical documents. With these images were investigated the nine methods to remove back-to-front interference: Isodata, Pun, Kapur-Sahoo-Wong, Johannsen-Bille, Yen-Chang-Chang, Otsu, Mello-Lins, Silva-Lins-Rocha and Wu-Lu. At the end, 1,600 binarized images and 1,600 matrices of co-occurrence probability were evaluated, also considering the visual inspection of images, and discusses the quality assessment of images binary.

From the generated database, with information from $1,600$ matrices of co-occurrence probability, one can find the best filter for different textures of historical documents with front-to-back interference. As an example, for the condition of 1.00% pixel loss in the text and 1.00% front-to-back interference tolerance, one has $P(f|f) = 99.00\%$ and $P(b|b) = 99.00\%$. For each historical document, one can select a filter with the optimal threshold. The result is shown in tables 4.8 and 4.9:

Table 4.8: Result of the Proposed Filter with Synthetic Documents.

Red	Green	Blue	Grayscale	σ	Mode	σ-aging	alpha	p(b|b)	p(f|f)	Filter	Threshold	Channel
252	233	153	230	5	231	3	0.4	99.60	100.00	Proposed	126	Red
252	233	153	230	5	231	3	0.5	99.87	100.00	Proposed	126	Red
252	233	153	230	5	231	3	0.6	99.91	100.00	Proposed	126	Red
252	233	153	230	5	231	3	0.7	99.94	100.00	Proposed	126	Red
252	233	153	230	5	231	3	0.8	99.97	100.00	Proposed	126	Red
252	233	153	230	5	231	3	0.9	99.99	100.00	Proposed	126	Red
252	233	153	230	5	231	3	1.0	100.00	100.00	Proposed	126	Red
252	233	153	230	5	231	3	0.8	99.94	99.23	IsoData	137	
252	233	153	230	5	231	3	0.9	99.98	99.20	IsoData	137	
252	233	153	230	5	231	3	1.0	100.00	99.56	IsoData	138	
252	233	153	230	5	231	3	0.7	99.25	100.00	Kapur-Sahoo-Wong	147	
252	233	153	230	5	231	3	0.7	99.87	99.54	Otsu	138	
252	233	153	230	5	231	3	0.8	99.94	99.56	Otsu	138	
252	233	153	230	5	231	3	0.9	99.97	99.53	Otsu	138	
252	233	153	230	5	231	3	1.0	99.95	99.56	Otsu	140	
252	233	153	230	5	231	3	0.8	98.98	100.00	Silva-Lins-Rocha	161	
252	233	153	230	5	231	3	0.9	99.07	100.00	Silva-Lins-Rocha	167	
252	233	153	230	5	231	3	1.0	99.26	100.00	Silva-Lins-Rocha	165	
252	233	153	230	5	231	3	0.9	99.19	100.00	Mello-Lins	165	
253	238	158	233	3	233	3	0.4	99.24	100.00	Proposed	130	Red
253	238	158	233	3	233	3	0.5	99.75	100.00	Proposed	130	Red
253	238	158	233	3	233	3	0.6	99.80	100.00	Proposed	130	Red
253	238	158	233	3	233	3	0.7	99.82	100.00	Proposed	130	Red
253	238	158	233	3	233	3	0.8	99.85	100.00	Proposed	130	Red
253	238	158	233	3	233	3	0.9	99.87	100.00	Proposed	130	Red
253	238	158	233	3	233	3	1.0	99.89	100.00	Proposed	130	Red
253	238	158	233	3	233	3	0.7	99.75	99.45	Otsu	141	
253	238	158	233	3	233	3	0.8	99.92	99.16	Otsu	140	
253	238	158	233	3	233	3	0.9	99.97	99.14	Otsu	140	
253	238	158	233	3	233	3	0.9	99.04	100.00	Silva-Lins-Rocha	171	
253	238	158	233	3	233	3	0.9	99.29	100.00	Mello-Lins	165	
241	200	136	205	4	206	5	0.4	99.50	100.00	Proposed	116	Red
241	200	136	205	4	206	5	0.5	99.84	100.00	Proposed	116	Red
241	200	136	205	4	206	5	0.6	99.88	100.00	Proposed	116	Red
241	200	136	205	4	206	5	0.7	99.91	100.00	Proposed	116	Red
241	200	136	205	4	206	5	0.8	99.94	100.00	Proposed	116	Red
241	200	136	205	4	206	5	0.9	99.96	100.00	Proposed	116	Red
241	200	136	205	4	206	5	1.0	99.97	100.00	Proposed	116	Red
241	200	136	205	4	206	5	0.7	99.19	99.53	IsoData	122	
241	200	136	205	4	206	5	0.8	99.84	99.55	IsoData	120	
241	200	136	205	4	206	5	0.9	99.87	99.55	IsoData	121	
241	200	136	205	4	206	5	0.8	99.79	99.55	Otsu	122	
241	200	136	205	4	206	5	0.9	99.81	99.55	Otsu	123	
241	200	136	205	4	206	5	0.9	99.04	100.00	Silva-Lins-Rocha	147	
245	204	141	209	4	211	5	0.4	99.50	100.00	Proposed	120	
245	204	141	209	4	211	5	0.5	99.84	100.00	Proposed	120	
245	204	141	209	4	211	5	0.6	99.88	100.00	Proposed	120	
245	204	141	209	4	211	5	0.7	99.91	100.00	Proposed	120	
245	204	141	209	4	211	5	0.8	99.94	100.00	Proposed	120	
245	204	141	209	4	211	5	0.9	99.96	100.00	Proposed	120	
245	204	141	209	4	211	5	1.0	99.97	100.00	Proposed	120	
245	204	141	209	4	211	5	0.7	99.59	99.60	IsoData	124	
245	204	141	209	4	211	5	0.8	99.87	99.65	IsoData	124	
245	204	141	209	4	211	5	0.9	99.92	99.64	IsoData	124	
245	204	141	209	4	211	5	0.7	99.19	99.60	Otsu	127	
245	204	141	209	4	211	5	0.8	99.84	99.65	Otsu	125	
245	204	141	209	4	211	5	0.9	99.87	99.64	Otsu	126	
245	204	141	209	4	211	5	0.9	99.04	100.00	Silva-Lins-Rocha	152	
201	181	134	182	7	184	7	0.4	99.50	99.41	Proposed	78	Red
201	181	134	182	7	184	7	0.5	99.84	99.40	Proposed	78	Red
201	181	134	182	7	184	7	0.6	99.88	99.41	Proposed	78	Red
201	181	134	182	7	184	7	0.7	99.91	99.42	Proposed	78	Red
201	181	134	182	7	184	7	0.8	99.94	99.42	Proposed	78	Red
201	181	134	182	7	184	7	0.9	99.96	99.42	Proposed	78	Red
201	181	134	182	7	184	7	1.0	99.97	99.42	Proposed	78	Red
201	181	134	182	7	184	7	0.9	99.09	100.00	Silva-Lins-Rocha	127	
224	174	113	182	4	182	5	0.4	99.60	100.00	Proposed	98	Red
224	174	113	182	4	182	5	0.5	99.87	100.00	Proposed	98	Red
224	174	113	182	4	182	5	0.6	99.91	100.00	Proposed	98	Red
224	174	113	182	4	182	5	0.7	99.94	100.00	Proposed	98	Red
224	174	113	182	4	182	5	0.8	99.97	100.00	Proposed	98	Red
224	174	113	182	4	182	5	0.9	99.99	100.00	Proposed	98	Red
224	174	113	182	4	182	5	1.0	100.00	100.00	Proposed	98	Red
254	247	208	244	4	246	5	0.4	99.60	100.00	Proposed	121	Green
254	247	208	244	4	246	5	0.5	99.87	100.00	Proposed	121	Green
254	247	208	244	4	246	5	0.6	99.91	100.00	Proposed	121	Green
254	247	208	244	4	246	5	0.7	99.94	100.00	Proposed	121	Green
254	247	208	244	4	246	5	0.8	99.97	100.00	Proposed	121	Green
254	247	208	244	4	246	5	0.9	99.99	100.00	Proposed	121	Green
254	247	208	244	4	246	5	1.0	100.00	100.00	Proposed	121	Green
254	247	208	244	4	246	5	0.7	99.48	100.00	Kapur-Sahoo-Wong	164	
254	247	208	244	4	246	5	0.8	99.17	100.00	Kapur-Sahoo-Wong	180	
254	247	208	244	4	246	5	0.9	99.04	100.00	Silva-Lins-Rocha	126	
254	247	208	244	4	246	5	1.0	99.27	100.00	Silva-Lins-Rocha	123	
254	247	208	244	4	246	5	0.8	99.67	100.00	Mello-Lins	168	
254	247	208	244	4	246	5	0.9	99.84	100.00	Mello-Lins	164	
254	247	208	244	4	246	5	1.0	99.45	100.00	Mello-Lins	179	

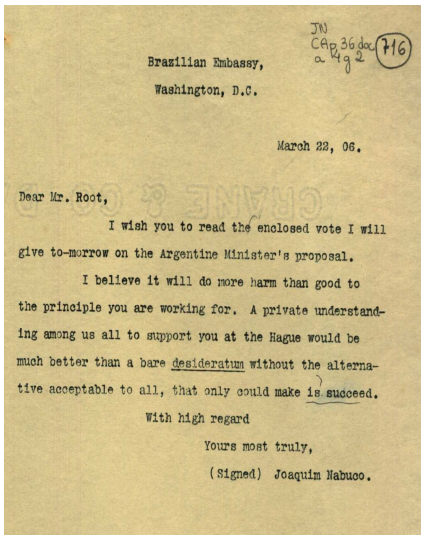
4.6 Results of the Proposed Method

In the previous section the results for the synthesized images were presented. Now, in the Figures 4.8 and 4.9 the results applied to historical documents.

(a) Historical Document 1. (b) Binarized Historical Document 1.

(c) Historical Document 2. (d) Binarized Historical Document 2.

Figure 4.8: Result of the Proposed Filter with Historical Documents.



Brazilian Embassy,
Washington, D.C.

March 22, 06.

Dear Mr. Root,

I wish you to read the enclosed vote I will give to-morrow on the Argentine Minister's proposal.

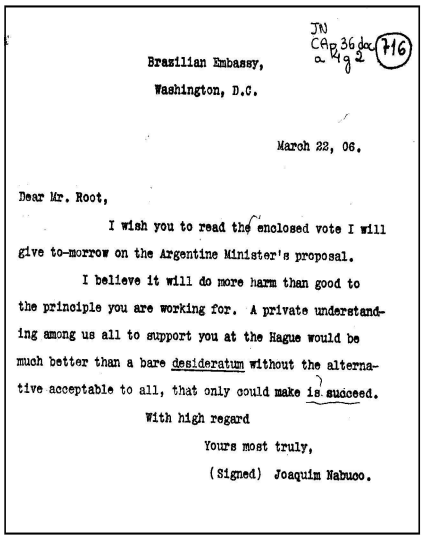
I believe it will do more harm than good to the principle you are working for. A private understanding among us all to support you at the Hague would be much better than a bare desideratum without the alternative acceptable to all, that only could make is succeed.

With high regard

Yours most truly,

(Signed) Joaquim Nabuco.

(a) Historical Document 3.



Brazilian Embassy,
Washington, D.C.

March 22, 06.

Dear Mr. Root,

I wish you to read the enclosed vote I will give to-morrow on the Argentine Minister's proposal.

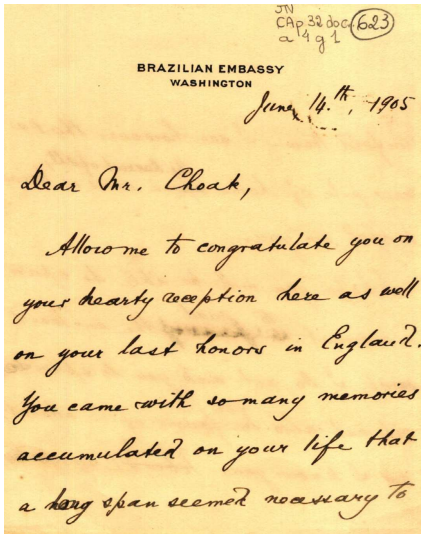
I believe it will do more harm than good to the principle you are working for. A private understanding among us all to support you at the Hague would be much better than a bare desideratum without the alternative acceptable to all, that only could make is succeed.

With high regard

Yours most truly,

(Signed) Joaquim Nabuco.

(b) Binarized Historical Document 3.



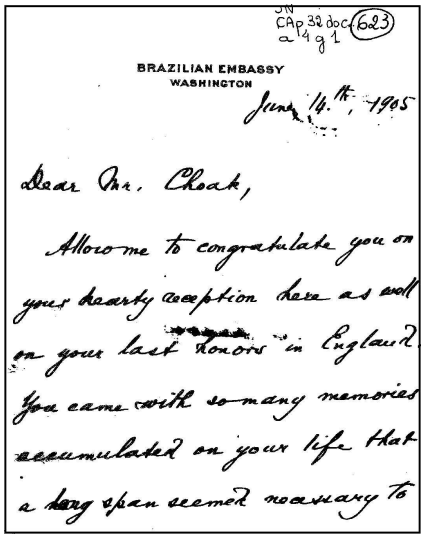
BRAZILIAN EMBASSY
WASHINGTON

June 14th, 1905

Dear Mr. Choate,

Allow me to congratulate you on your hearty reception here as well on your last honors in England. You came with so many memories accumulated on your life that a long span seemed necessary to

(c) Historical Document 4.



BRAZILIAN EMBASSY
WASHINGTON

June 14th, 1905

Dear Mr. Choate,

Allow me to congratulate you on your hearty reception here as well on your last honors in England. You came with so many memories accumulated on your life that a long span seemed necessary to

(d) Binarized Historical Document 4.

Figure 4.9: Result of the Proposed Filter with Historical Documents.

4.7 Conclusions

The filter proposed for back-to-front interference removal when tested with the 160 synthetic documents, yielded binary images that meet the imposed conditions of the co-occurrence probabilities $P(b|b) \geq 99.00\%$ and P $(f|f) \geq 99.00\%$ shown as an example in Section 4.5.

The visual inspection of the binarized images studied confirmed the results obtained for the matrix co-occurrence probability.

Two of the test documents, the ones with texture (R=224, G=174, B=113) and (R=218, G=167, B=113), only provided reasonable results when binarized by the proposed filter, given the efficiency requirements imposed that the error is less of 1.00%

In each binarized document one has one or more filters selected with the optimum threshold and corresponding co-occurrence probabilities $P(b|b)$ and $(f|f)$, providing a measure of the quality of the binarized images. The remaining two historical documents the filters were not able to meet the above conditions.

The filters that met the requirements of co-occurrence probabilities in the given example were: the Proposed Filter, Isodata, Otsu, Kapur-Sahoo-Wong, Silva-Lins-Rocha and the Mello-Lins. The opacity factor value α varied from 0.4 to 1.0, because of the inclusion of the proposed filter in the inclusion to remove back-to-front interference and the binarization processes.

Table 4.9: Continuation of Table 4.8.

Red	Green	Blue	Grayscale	σ	Mode	σ-aging	alpha	p(b|b)	p(f|f)	Filter	Threshold	Channel
253	232	152	229	4	229	3	0.4	99.60	100.00	Proposed	127	Red
253	232	152	229	4	229	3	0.5	99.87	100.00	Proposed	127	Red
253	232	152	229	4	229	3	0.6	99.91	100.00	Proposed	127	Red
253	232	152	229	4	229	3	0.7	99.94	100.00	Proposed	127	Red
253	232	152	229	4	229	3	0.8	99.97	100.00	Proposed	127	Red
253	232	152	229	4	229	3	0.9	99.99	100.00	Proposed	127	Red
253	232	152	229	4	229	3	1.0	100.00	100.00	Proposed	127	Red
253	232	152	229	4	229	3	0.7	99.78	99.21	IsoData	136	
253	232	152	229	4	229	3	0.8	99.92	99.21	IsoData	136	
253	232	152	229	4	229	3	0.9	99.97	99.20	IsoData	136	
253	232	152	229	4	229	3	1.0	100.00	99.51	IsoData	137	
253	232	152	229	4	229	3	0.8	99.07	100.00	Kapur-Sahoo-Wong	157	
253	232	152	229	4	229	3	0.7	99.59	99.53	Otsu	139	
253	232	152	229	4	229	3	0.8	99.87	99.54	Otsu	139	
253	232	152	229	4	229	3	0.9	99.92	99.52	Otsu	139	
253	232	152	229	4	229	3	1.0	99.95	99.51	Otsu	139	
253	232	152	229	4	229	3	0.9	99.04	100.00	Silva-Lins-Rocha	167	
253	232	152	229	4	229	3	1.0	99.27	100.00	Silva-Lins-Rocha	164	
253	232	152	229	4	229	3	0.9	99.14	100.00	Mello-Lins	165	
218	167	113	177	6	180	5	0.4	99.60	100.00	Proposed	92	Red
218	167	113	177	6	180	5	0.5	99.87	100.00	Proposed	92	Red
218	167	113	177	6	180	5	0.6	99.91	100.00	Proposed	92	Red
218	167	113	177	6	180	5	0.7	99.94	100.00	Proposed	92	Red
218	167	113	177	6	180	5	0.8	99.97	100.00	Proposed	92	Red
218	167	113	177	6	180	5	0.9	99.99	100.00	Proposed	92	Red
218	167	113	177	6	180	5	1.0	100.00	100.00	Proposed	92	Red
253	223	156	224	6	227	5	0.4	99.60	100.00	Proposed	127	Red
253	223	156	224	6	227	5	0.5	99.87	100.00	Proposed	127	Red
253	223	156	224	6	227	5	0.6	99.91	100.00	Proposed	127	Red
253	223	156	224	6	227	5	0.7	99.94	100.00	Proposed	127	Red
253	223	156	224	6	227	5	0.8	99.97	100.00	Proposed	127	Red
253	223	156	224	6	227	5	0.9	99.99	100.00	Proposed	127	Red
253	223	156	224	6	227	5	1.0	100.00	100.00	Proposed	127	Red
253	223	156	224	6	227	5	1.0	100.00	99.09	IsoData	135	
253	223	156	224	6	227	5	0.8	99.07	100.00	Kapur-Sahoo-Wong	156	
253	223	156	224	6	227	5	0.7	99.75	99.45	Otsu	136	
253	223	156	224	6	227	5	0.8	99.92	99.14	Otsu	136	
253	223	156	224	6	227	5	0.9	99.97	99.46	Otsu	136	
253	223	156	224	6	227	5	1.0	99.95	99.48	Otsu	138	
253	223	156	224	6	227	5	0.9	99.04	100.00	Silva-Lins-Rocha	166	
253	223	156	224	6	227	5	1.0	99.27	100.00	Silva-Lins-Rocha	163	
253	223	156	224	6	227	5	0.9	99.09	100.00	Mello-Lins	165	
248	197	132	205	6	207	7	0.4	99.60	100.00	Proposed	124	Red
248	197	132	205	6	207	7	0.5	99.87	100.00	Proposed	124	Red
248	197	132	205	6	207	7	0.6	99.91	100.00	Proposed	124	Red
248	197	132	205	6	207	7	0.7	99.94	100.00	Proposed	124	Red
248	197	132	205	6	207	7	0.8	99.97	100.00	Proposed	124	Red
248	197	132	205	6	207	7	0.9	99.99	100.00	Proposed	124	Red
248	197	132	205	6	207	7	1.0	100.00	100.00	Proposed	124	Red
248	197	132	205	6	207	7	1.0	100.00	99.09	IsoData	124	
248	197	132	205	6	207	7	0.8	99.07	100.00	Kapur-Sahoo-Wong	140	
248	197	132	205	6	207	7	0.7	99.75	99.45	Otsu	132	
248	197	132	205	6	207	7	0.8	99.92	99.14	Otsu	125	
248	197	132	205	6	207	7	0.9	99.97	99.46	Otsu	126	
248	197	132	205	6	207	7	1.0	99.95	99.48	Otsu	126	
248	197	132	205	6	207	7	0.9	99.04	100.00	Silva-Lins-Rocha	149	
248	197	132	205	6	207	7	1.0	99.27	100.00	Silva-Lins-Rocha	146	
248	197	132	205	6	207	7	0.9	99.09	100.00	Mello-Lins	165	
212	192	141	192	7	193	7	0.4	99.66	99.49	Proposed	87	Red
212	192	141	192	7	193	7	0.5	99.87	99.49	Proposed	87	Red
212	192	141	192	7	193	7	0.6	99.91	99.49	Proposed	87	Red
212	192	141	192	7	193	7	0.7	99.94	99.49	Proposed	87	Red
212	192	141	192	7	193	7	0.8	99.97	99.49	Proposed	87	Red
212	192	141	192	7	193	7	0.9	99.99	99.49	Proposed	87	Red
212	192	141	192	7	193	7	1.0	100.00	99.43	Proposed	87	Red
212	192	141	192	7	193	7	0.7	99.34	99.55	IsoData	121	
212	192	141	192	7	193	7	1.0	99.91	99.58	IsoData	112	
212	192	141	192	7	193	7	0.8	99.79	99.56	Otsu	113	
212	192	141	192	7	193	7	0.9	99.84	99.58	Otsu	113	
212	192	141	192	7	193	7	1.0	99.88	99.58	Otsu	113	
212	192	141	192	7	193	7	0.9	99.09	100.00	Silva-Lins-Rocha	137	
212	192	141	192	7	193	7	1.0	99.30	100.00	Silva-Lins-Rocha	134	
242	243	247	243	5	246	5	0.4	99.60	100.00	Proposed	116	Red
242	243	247	243	5	246	5	0.5	99.87	100.00	Proposed	116	Red
242	243	247	243	5	246	5	0.6	99.91	100.00	Proposed	116	Red
242	243	247	243	5	246	5	0.7	99.94	100.00	Proposed	116	Red
242	243	247	243	5	246	5	0.8	99.97	100.00	Proposed	116	Red
242	243	247	243	5	246	5	0.9	99.99	100.00	Proposed	116	Red
242	243	247	243	5	246	5	1.0	100.00	100.00	Proposed	116	Red
242	243	247	243	5	246	5	0.7	99.59	100.00	Kapur-Sahoo-Wong	171	
242	243	247	243	5	246	5	0.8	99.17	100.00	Kapur-Sahoo-Wong	188	
242	243	247	243	5	246	5	0.9	99.04	100.00	Silva-Lins-Rocha	199	
242	243	247	243	5	246	5	1.0	99.27	100.00	Silva-Lins-Rocha	196	
242	243	247	243	5	246	5	0.7	99.34	100.00	Mello-Lins	173	
242	243	247	243	5	246	5	0.8	99.89	100.00	Mello-Lins	170	
242	243	247	243	5	246	5	1.0	99.67	100.00	Mello-Lins	182	
226	219	207	220	4	219	3	0.4	99.60	100.00	Proposed	100	Red channel
226	219	207	220	4	219	3	0.5	99.87	100.00	Proposed	100	Red channel
226	219	207	220	4	219	3	0.6	99.91	100.00	Proposed	100	Red channel
226	219	207	220	4	219	3	0.7	99.94	100.00	Proposed	100	Red channel
226	219	207	220	4	219	3	0.8	99.97	100.00	Proposed	100	Red channel
226	219	207	220	4	219	3	0.9	99.99	100.00	Proposed	100	Red channel
226	219	207	220	4	219	3	1.0	100.00	100.00	Proposed	100	Red channel
226	219	207	220	4	219	3	0.7	99.48	100.00	Kapur-Sahoo-Wong	145	
226	219	207	220	4	219	3	0.8	99.17	100.00	Kapur-Sahoo-Wong	161	
226	219	207	220	4	219	3	0.9	99.04	100.00	Silva-Lins-Rocha	172	
226	219	207	220	4	219	3	1.0	99.27	100.00	Silva-Lins-Rocha	169	
226	219	207	220	4	219	3	0.9	99.29	100.00	Mello-Lins	166	

5 BINARIZING COMPLEX SCANNED DOCUMENTS

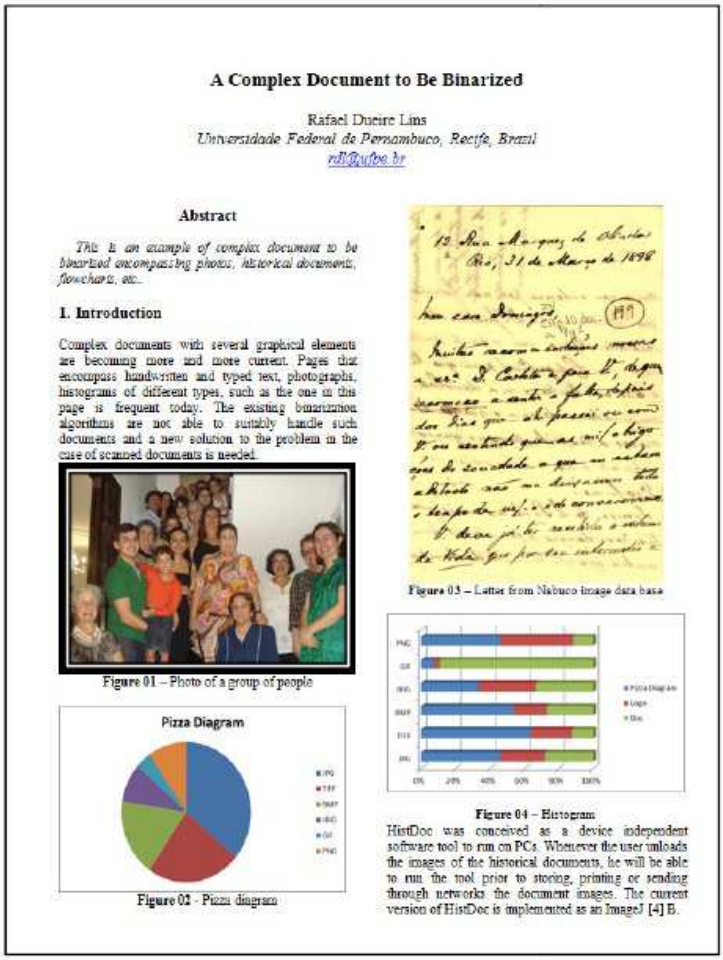
Most binarization algorithms are suitable for scanned text documents and do not work adequately with complex documents that encompass photos, charts and text simultaneously. This chapter presents a new binarization algorithm that works on complex documents. Each of the elements in the image is processed depending on its nature, thus their binarization takes that into account to preserve the original content.

5.1 Introduction

Complex documents encompassing not only text but also several graphic elements such as photos, histograms, pie (or pizza) diagrams, etc. are becoming of widespread use. Figure 5.1 shows an example of such a document. The binarization algorithms found in the literature are not suitable for such documents, as most of them use global threshold techniques.

The direct binarization of Figure 5.1 using (Otsu, 1979) algorithm provides the image shown in Figure 5.2, in which one may observe that the information provided by the pictorial elements was completely lost and that even the historic letter shown with back-to-front interference lost its legibility after binarization.

In 2007, the authors of this paper in the LiveMemory project (Lins, 2010a) took the challenge of building a digital library with the ten previous years of the proceedings of the Brazilian Telecommunications Society and distributing it with all the participants of the event in a DVD. In 2010, they had finished the library encompassing 26 years of the events. For that task 12 years of proceedings had to be scanned, filtered, and indexed. Pages were originally scanned in true color, 300 d.p.i. resolution. Back-to-front interference (Monte da Silva *et al.*, 2008) (bleeding) was observed in many pages and had to be filtered out. As the number of images to be included in the DVD was far too big, images had to be binarized. Several pages included graphical elements as photos, graphs, bar histograms, pie diagrams, etc. The direct binarization of such pages yielded the complete loss of the information of the graphical elements, in a similar situation to


A Complex Document to Be Binarized

Rafael Dueire Lins
Universidade Federal de Pernambuco, Recife, Brazil
rdl@ufpe.br

Abstract

This is an example of complex document to be binarized encompassing photos, historical documents, flowcharts, etc..

1. Introduction

Complex documents with several graphical elements are becoming more and more current. Pages that encompass handwritten and typed text, photographs, histograms of different types, such as the one in this page is frequent today. The existing binarization algorithms are not able to suitably handle such documents and a new solution to the problem in the case of scanned documents is needed.

Figure 01 – Photo of a group of people

Pizza Diagram

Figure 02 - Pizza diagram

Figure 03 – Letter from Nabuco image data base

Figure 04 – Histogram

HistDoc was conceived as a device independent software tool to run on PCs. Whenever the user unloads the images of the historical documents, he will be able to run the tool prior to storing, printing or sending through networks the document images. The current version of HistDoc is implemented as an ImageJ [4] B.

Figure 5.1: Example of complex document encompassing text and several graphic elements.

the one shown in Figure 5.2. The solution adopted then was to disassemble such pages, binarize the text area and re-assemble the page with the graphical elements in hues of grey. Such pages claimed far more storage space than the "real" binary ones, but such trick of binarizing the text area and background provided the reader a "continuity effect" as such pages did not differ much from the text-only (binary) pages. Figure 5.3 shows the page shown in Figure 5.1 processed using the LiveMemory scheme (Lins, 2010a).

The data presented in Table 5.1 shows that the LiveMemory processing scheme provides a good image quality and size tradeoff. As the text area is black and all the paper background area is plain white the compressed version of the document is much smaller than the gray-scale equivalent image. The size difference between the LiveMemory synthetic image and the binary (monochromatic) one is a factor of at least 6 times for GIF and 11 times for PNG.

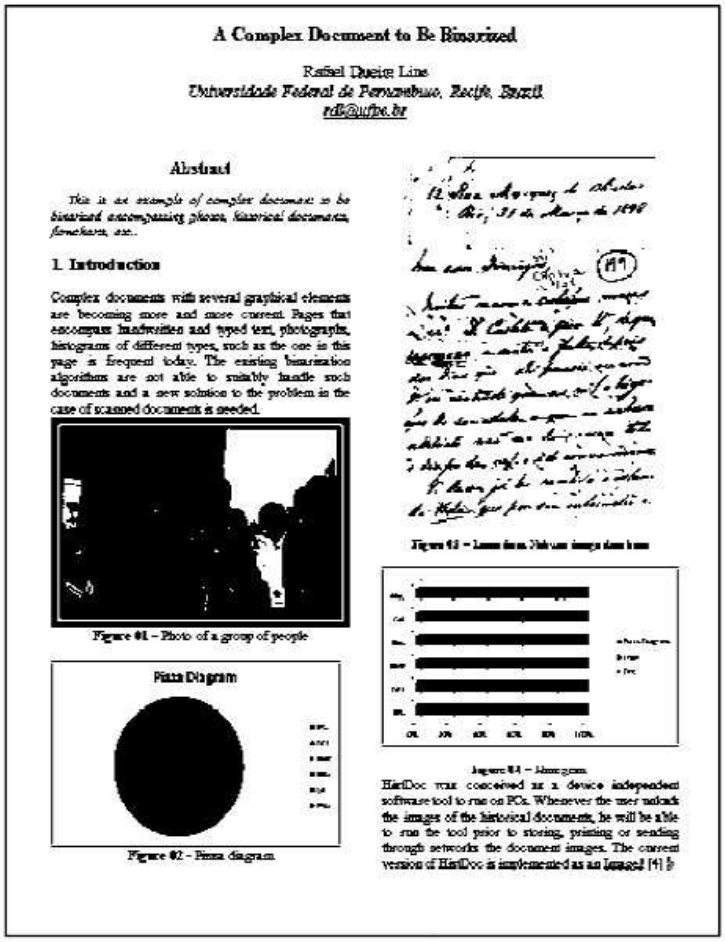


Figure 5.2: Result of the direct binarization of the document presented in Figure 5.2 using Otsu global binarization algorithm.

Table 5.1: The size in Kbytes of the image in Figure 1.

Kbytes	Bitmap	PNG	TIFF	GIF
True-color	25,509	7,704	8,997	
Gray-scale	8,514	3,461	8,503	903
LiveMemory		1,221	1,552	858
B&W	1,07	104	1,066	130

This work presents a binarization scheme suitable for treating scanned complex documents. The scheme automatically decomposes a document image and identifies each of the graphical elements to provide a suitable binarization for each of them independently. In the binarization phase of all the elements, the document image is re-assembled to yield the binary original document. Camera acquired documents have degree of complexity (Silva & Lin, 2007) and are not addressed in this image. The extra complexity is due to uneven illumination that most cameras store images in JPEG file format, a scheme that

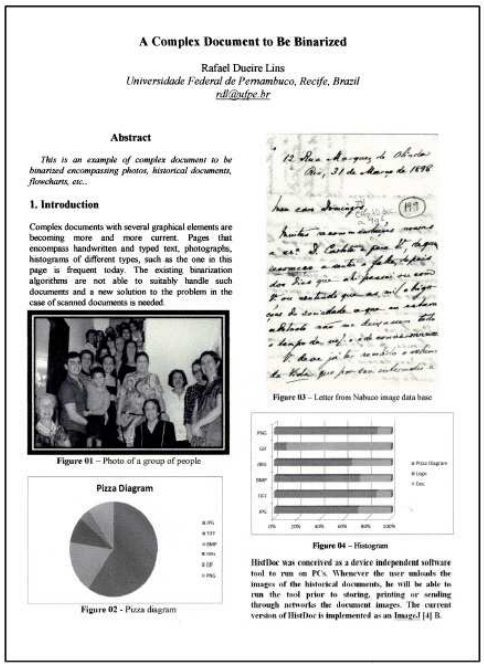


Figure 5.3: LiveMemory processing scheme used in the document shown in Figure 5.1.

introduces losses and the JPEG artifact assumes that the document to be processed was scanned in 300 d.p.i resolution and stored in a lossless file format, with no sort of noises (physical, digitalization, storage, etc.) as described reference (Lins, 2009b).

5.2 The New Binarization Scheme

Image de-skew was performed here using the algorithm presented in reference Avila & Lins (2005) and its purpose is to remove skew that may have been introduced during document scanning. Projection profile is a standard technique for spotting image blocks in such kind of document. As projection profile is sensitive to any document skew, the previous step is fundamental for the accuracy of the results of block determination. Block classification is the next step, which is detailed later on. Then each block is independently binarized. Finally, the document is re-assembled with the results of the binary blocks. The operations enumerated above are schematically depicted in Figure 5.4. It is important to say that, although Figure 5.4 presents as if the projection profile and block spotting were performed on the true color version of the image, it is really performed on top of the binary version of documents, obtained by a straightforward binarization process using a

threshold algorithm such as Otsu, producing black blocks as the ones presented in Figure 5.2.

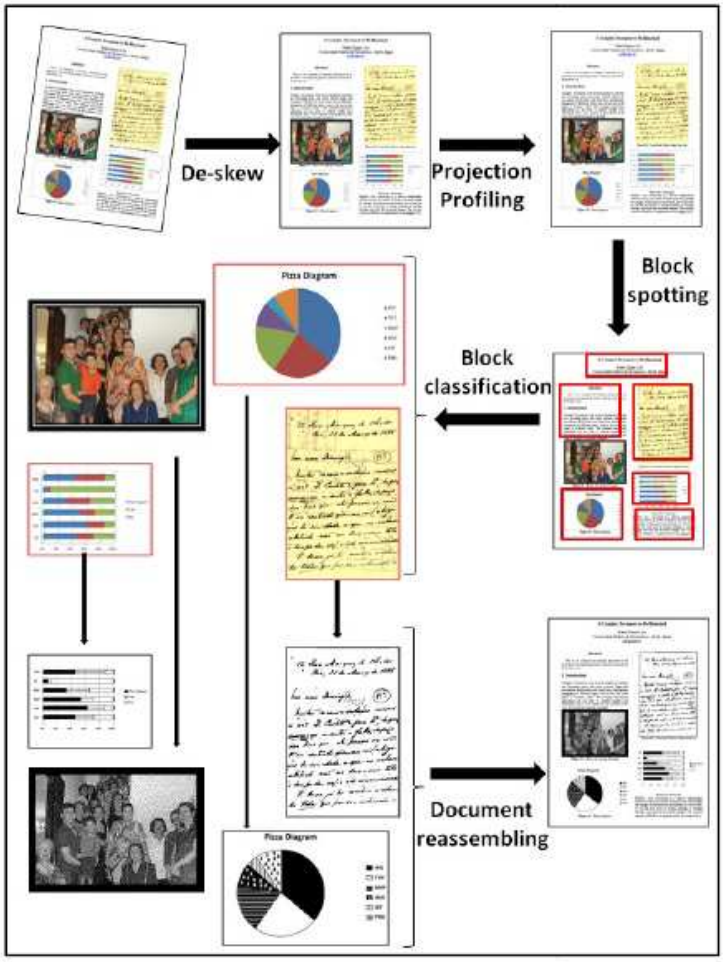


Figure 5.4: Binarization scheme proposed.

The main principle of the scheme proposed here is the recognition that each of the blocks in the complex document has a different nature, and no binarization algorithm that does not take such information into account has the slightest chance of succeeding in keeping the fundamental information of the original document.

5.3 The Block Classifier

The block classifier developed here was based on the one developed by the authors together with the Hewlett-Packard researchers for improving the quality of image printing Lins (2009a), which on its turn is based on the binary classification approach originally

described in Simske (2005). It assumes a Gaussian distribution for each of the features, and its performance degrades in proportion to the non-Gaussian nature of the data. The kinds of graphical elements analyzed in this classification level were:

1. Printed text (PT)
2. Cursive text (CT)
3. Photo (Ph)
4. Logo (Lg)
5. Pie or Pizza diagrams (Pie)
6. Histograms or bar diagrams (Hist)
7. Compound complex block/Don't know (Cpd)

The basis of the binarization scheme proposed here is to split the document to be binarized into different blocks, analyze and classify each of them and binarize each of them depending of their nature. The binarization scheme performs thus the following steps:

1. Image de-skew;
2. Projection profile;
3. Block spotting;
4. Block classification;
5. Block binarization;
6. Document re-assembling.

classification:

- Palette (true-color/grayscale).
- Gamut.
- Conversion into Grayscale (if RGB).

- Gamut in Grayscale (if RGB).
- Conversion into Binary (Otsu).
- Mean edge value.
- Number of black pixels in binary image.
- (#Black pixels/Total # pixels)*100%.
- (Gamut/Palette)*100% (true/grayscale).

The classifier works as a set of cascaded random forest binary classifiers as shown in Figure 5.5.

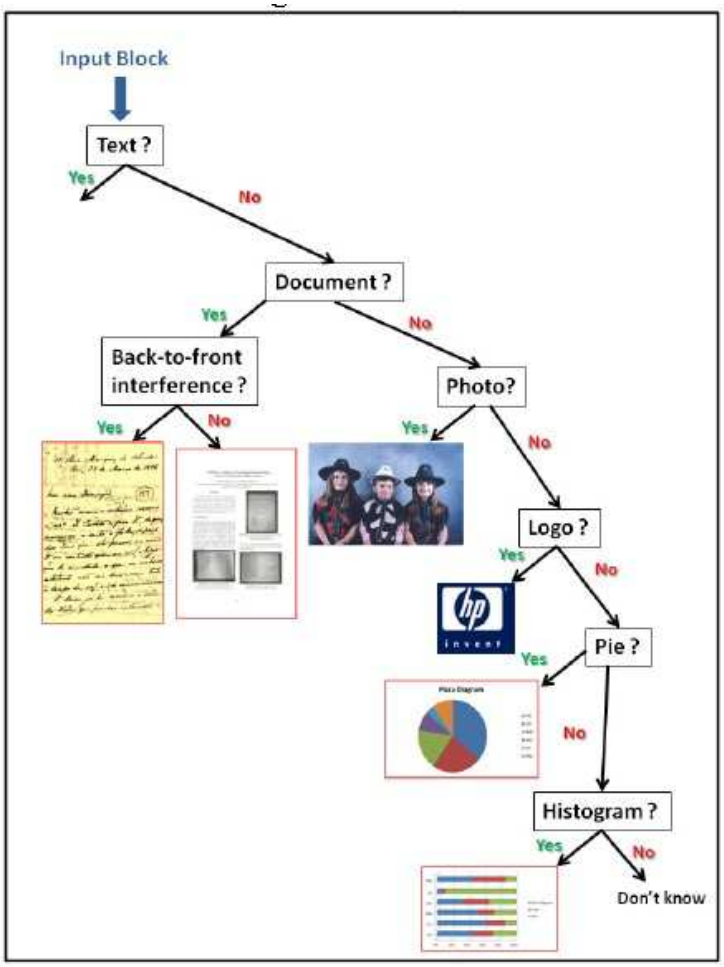


Figure 5.5: Binarization scheme proposed.

The classifier was trained and tested with images of each of the seven types totaling

30, 300 images (Cpd= 300 images others= 5, 000). The confusion matrix obtained using cross-validation with k-folders= 10 is shown in Table 5.2.

Table 5.2: Confusion Matrix for the random forest cascaded classifier.

	Printed text	**Cursive text**	**Photo**	**Logo**	**Pie**	**Histograms**	**Compound complex**
Printed text	4,966	29	0	4	0	0	1
Cursive text	37	4,958	0	0	0	2	3
Photo	0	0	4,961	36	1	0	2
Logo	0	0	48	4,909	16	15	12
Pie	0	0	3	20	4,971	4	2
Histograms	0	0	0	31	18	4,95	1
Compound complex	0	0	8	2	1	0	289

The classification figures presented in the confusion matrix presented in Table 5.2 shows that the classifier proposed provides good results, yielding a precision of 0.99 for most images and drops down to 0.93 in the case of compound/complex ones Figure 5.6 presents an example of a Compound complex Block, which corresponds to a Brazilian Identification Card, encompassing a photo, a background printed area, stamp, fingerprint on white background, and signature. The direct binarization of such an image, similarly to the image of the document page shown in Figure 5.2 represents a complete loss of the original information. The binarization of such complex block image is left out of the scope of this paper.

Figure 5.6: Compound complex block encompassing different graphical elements and its binarization using Otsu algorithm.

5.4 Binarizing Photos

As already presented, the direct binarization of photos does not yield an image capable of providing the minimum possible recognition. Image dithering offers a reasonable solution to the problem. Helland (2014) provides a brief explanation and source code for 11 image dithering algorithms. Floyd & Steinberg (1976) dithering is easily the most well-known error diffusion algorithm. It provides reasonably good quality, while only requiring a single forward array (a one-dimensional array with the width of the image, which stores the error values pushed to the next row). Additionally, because its divisor is 16, bit-shifting can be used in place of division making it quite fast, even on old hardware. Figure 5.7 zooms into the photo presented in Figure 5.1. The direct application of Floyd-Steinberg algorithm to it yielded the monochromatic image presented in Figure 5.8 in which one may observe that most of the information of the original content was preserved.

The binary photo presented in Figure 5.7 has over 75% of its area of relevant information, exhibiting very little in background. Most of times, there is no relevant information in the background that is converted into noise in the binary image. The goal is to find a new "intelligent" dithering algorithm that takes into account the content of the photo to be binarized. In such a way, the photo will be classified as people, landscape, object, and complex. Whenever a photo is recognized as being of an object the background is completely removed using an algorithm similar to the one described in reference Lins & Silva (2013) prior to dithering.

Figure 5.7: Photo of a group of people presented in the document page in Figure 5.1.

Figure 5.8: Photo of a group of people presented in the document page in Figure 5.1.

5.5 Binarizing Logos

Logos tend to use a few plain colors. The binarization of logos may be a straightforward process in the case of single color logos or may claim for a more complex strategy in the case of using multiple colors. Figure 09 presents some logos and their direct binarization using Otsu algorithm.

The five logos presented in Figure show that the direct binarization using Otsu algorithm yields results that preserve the original information in most cases (the logo of

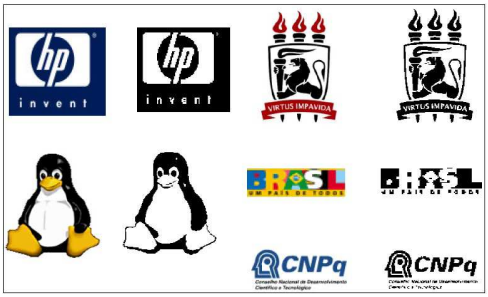


Figure 5.9: Five different logos and their binarization using Otsu algorithm.

"BRASIL" has un unusual number of colors). It is important to remark that the block splitting strategy used here was important to provide the satisfactory results shown. The use of a global threshold binarization algorithm if applied to the whole document page may provide a different threshold value that may destroy part of the information.

5.6 Binarizing Documents

The binarization of text documents either printed or handwritten has a long tradition in document engineering. Document aging, back-to-front interference (bleeding) and some physical noises Lins (2009b) may bring an extra degree of difficulty to this task. In this work, the algorithm described in reference Monte da Silva *et al.* (2008) was used to process the handwritten historical document shown. The authors are currently working on a document classifier that refines the different kinds of documents, making possible to better select and tune the binarization algorithm for documents.

5.7 Binarizing Pie Diagrams and Histograms

Pie (or Pizza) diagrams and Histograms tend to be automatically generated by spreadsheet tools such as Microsoft Excel. As already shown, the direct binarization of such diagrams yield a complete loss of the original information. The only alternative is recognizing the different elements in the image and generating a synthetic monochromatic image that conveys the equivalent information. Different textures are used to represent the different colors in the diagram. In both kinds of diagrams the contour must be drawn first, together with the color separation areas. The largest slice in the pie is painted black,

the second largest is painted white, the third largest is replaced by a texture, etc.

5.8 Conclusions

The widespread use of electronic editing tools in the last three decades has yielded documents with several graphical elements incorporated. On the other hand, storage capacity in most organizations cannot meet the demand created and the binarization of such documents is still an option to save and distribute their information. No binarization algorithm is good enough to work suitably in all sorts of documents. This work proposed a new strategy to improve the binarization of complex documents that splits it into different blocks and classify them individually. Once the image blocks were suitably recognized and binarized using the most adequate strategy to the block nature, the image is reassembled yielding the monochromatic version of the original document.

The use of the scheme presented here applied in the document image shown in Figure 5.1 yielded the image shown in Figure 5.10 (the pie and histogram image blocks were enhanced manually because page scaling-down would not allow to distinguish the texture of the different areas). The size of the block assembled image is (in Kbytes): $1,858$ bmp, 132 tiff, 122 png, 58 gif. Notice that the new monochromatic image in gif is less than half of the size than the original image binarized directly using Otsu algorithm, but keeps the original information elements.

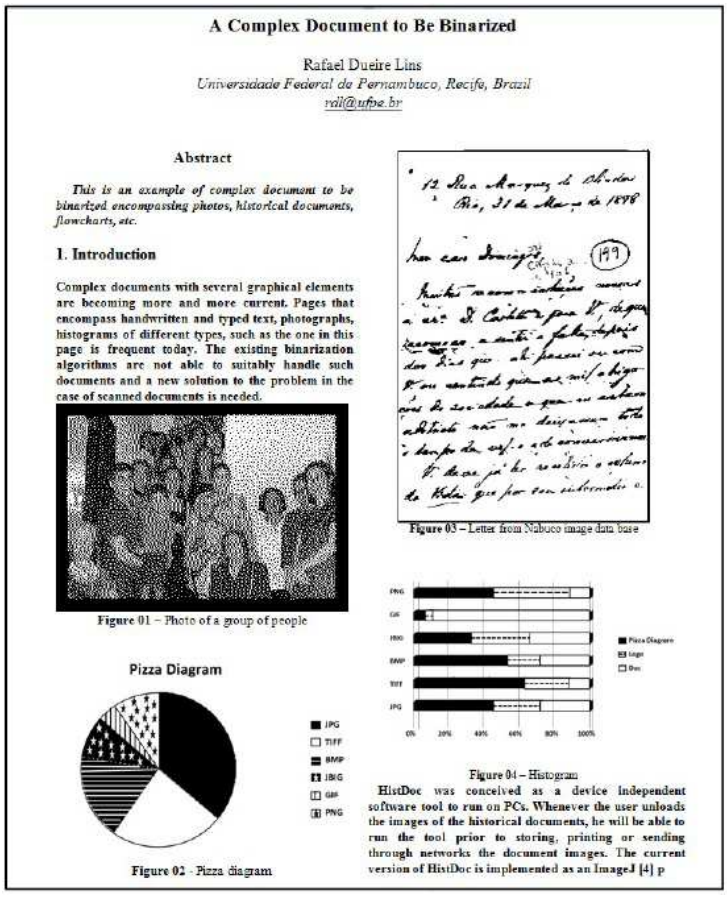


Figure 5.10: Final Block binarized image.

6 CONCLUSIONS AND LINES FOR FURTHER WORK

The contributions of this thesis are outlined here together with some research directions that may be followed as other unfolding of the results obtained.

Chapter 2 analyzed the consistency of the assessment for senior researchers in Computer Science of CNPq, the Brazilian Research Council. Several statistical classification strategies were considered using international public databases and were compared with the CNPq classification.

The most relevant results in such analysis listed in this thesis are:

- ArnetMiner and Google Scholar are the most representative databases for the senior researchers in Computer Science in Brazil, as they convey the largest number of publications per researcher.

- Taking as reference the ArnetMiner database, the average volume of publications by the researchers in the categories $1A$ and $1C$ from the Northeast is higher than the one in the other regions of Brazil.

- Using the discriminant analysis approach, and making a comparison between the various databases, we found that the data from the Google Scholar database has the highest proportion of correct hits as 85.2%, for the researchers in classifications $1A$, $1B$ and $1C$, although the number of samples is small.

- The analysis of the Lattes and ArnetMiner databases provides equivalent results of about 60.0%. The worst hit rate is obtained with the Web of Science database, 43.6%. The highest proportion of classification similarity is using the data provided by the Google Scholar database in the case of the researchers in the $1A$ group, 100.0%. The worst classification results is obtained with the Web of Science database in group $1C$, 20.6%.

- The discriminant analysis method was effective in identifying the appropriate database for the classification and ordering of researchers for $1A$, $1B$ and $1C$ levels, given the correct percentage of correlation to the CNPq classification. In this case, the Ar-

netMiner database represents the production of CNPq researchers correctly as well. Considering a set of all databases, the method remained robust with satisfactory results.

- When taking into consideration the geographic distribution of the researchers using the CNPq classification is uneven, with the Southeast region of Brazil with about 70.00% of all senior researchers, while the Notheast region represents only 7.00% of the total number of researchers. An important result obtained is that, the analysis of the scientific production, allows to say that overall there was **no discrimination** against the researchers in the regions with less senior researchers, and that one may even, contrary to the general belief, that some of the researchers from the less represented regions suffered *positive discrimination*, being ranked above their peers in the Southeast.

- The criteria presented by CNPq does not differ much from the results presented by the methods described, i.e., discriminant analysis and k-mean.

- With respect to the ArnetMiner database, it showed a consistency in the set of given variables, considering the volume of data as well as the representative variables able to generate a clustering and reclassification models.

As an overall conclusion, one way say that the CNPq rank is fair in general terms as the researcher classification distortions are small.

The Chapter 3 analyzed nine types of filters to binarize document images with different degrees of strength of the back-to-front interference, totaling 160 synthetic images as input. The output images generated a database of 1,440 binary documents. The matrix of co-occurrence probabilities and visual inspection were used to evaluate the quality of the resulting images, from which one may draw the following conclusions:

- Of the 16 historical documents analyzed, 12 could have some treatment by the studied filters, under the imposed conditions of the co-occurrence probabilities $P(b|b) \geq 99.00\%$ and P $(f|f) \geq 99.00\%$ shown as an example in Section 3.4. In each binarized document one has one or more filters selected with the optimum threshold

and corresponding co-occurrence probabilities $P(b|b)$ and $(f|f)$, providing a measure of the quality of the binarized images. Four historical documents were not able to meet the above conditions.

- The Pun method was not efficient in filtering the 160 images generated synthetically, because the co-occurrence probability $P(b|f)$ or maximum error was above 32.00%. The visual inspection of the binarized documents also corroborate to this result.

- With the visual inspection of the binarized images studied, as well as the matrix co-occurrence probability, one finds that the remaining algorithms are good thresholding methods to remove back-to-front interference in documents, depending on the texture of the historical document and the intensity of the back-to-front interference.

- The filters that met the requirements of co-occurrence probabilities in the given example were: Isodata, Otsu, Kapur-Sahoo-Wong, Silva-Lins-Rocha and Mello-Lins. The opacity factor value α varied from 0.7 to 1.0.

A new filter for back-to-front interference removal is proposed in Chapter 4. Tested with the 160 synthetic documents described in Chapter 3, it produced good-quality binary images that meet the imposed conditions of the co-occurrence probabilities $P(b|b) \geq$ 99.00% and P $(f|f) \geq$ 99.00%. The visual inspection of the binarized images studied confirmed the results obtained for the matrix co-occurrence probability.

Assessing the newly proposed filter together with the ones in Chapter 3 one may draw the following conclusions:

- Two of the test documents, the ones with texture (R=224, G=174, B=113) and (R=218, G=167, B=113), only provided reasonable results when binarized by the proposed filter, given the efficiency requirements imposed that the error is less of 1.00%

- For each binarized document one has one or more filters selected with the optimum threshold and corresponding co-occurrence probabilities $P(b|b)$ and $(f|f)$, providing a measure of the quality of the binarized images. The remaining two historical documents the filters were not able to meet the above conditions.

- The filters that met the requirements of co-occurrence probabilities in the given example were: the Proposed Filter, Isodata, Otsu, Kapur-Sahoo-Wong, Silva-Lins-Rocha and the Mello-Lins. The opacity factor value α varied from 0.4 to 1.0, because of the inclusion of the proposed filter in the inclusion to remove back-to-front interference and the binarization processes.

- The proposed filter performed better than all nine filters analyzed with respect to the alpha range, in some historical documents reaching the range of 0.4 to 1.0, when the co-occurrence probability P(b|b) and P(f|f) are above of 95.00%.

Chapter 5 proposes the "intelligent" binarization of complex documents, a document which encompasses several different graphical elements besides text, such as photos, logos, diagrams, pie charts, etc. This work proposes a new strategy to improve the binarization of complex documents that splits it into different blocks and classify them individually.

The basis of the *semantic binarization* scheme proposed here is to split the document to be binarized into different blocks, analyze and classify each of them and binarize each of them depending of their nature.

Some of the lines for further work along the lines of this thesis are:

- Refining the degree of the back-to-front intensity (α) in the critical regions to better "understand" the behavior of the different filters.

- In this work, the opacity factor α ranged from 0.1 to 1.00 in steps of 0.1. One alternative would be to do in smaller steps, especially around values where the of co-occurrence probability are more tolerable.

- Including new filters in the assessment.

- Consider other kinds of textures of historical documents.

- Developing several new algorithms for semantic binarization.

7 APPENDIX A

7.1 Bilateral Filtering for Gray and Color Images

The bilateral filter was first introduced by Aurich & Weule (1995) under the name "nonlinear Gaussian filter". It was rediscovered later by Tomasi & Manduchi (1998) who called it the "bilateral filter" which is now the most commonly used name according to Paris & Durand (2009). The bilateral filter is technique to smooth images while preserving edges. The filter output at each pixel is a weighted average of its neighbors. The weight assigned to each neighbor decreases with both the distance values among pixels of the image plane (the spatial domain S) and the distance on the intensity axis (the range domain R). The filter applies spatial weighted averaging without smoothing edges. The mathematical expression, Equation 7.1.7 is achieved by combining two Gaussian filters; one filter works in the spatial domain, the other filter works in the intensity domain. Therefore, not only the spatial distance but also the intensity distance is important for the determination of weights. At a pixel location (x, y), the output of a bilateral filter can be formulated as follows:

Bilateral filter combines two stages of filtering. These are the geometric closeness (i.e., filter domain) and the photometric similarity (i.e., filter range) among pixels in an $N \times N$ window size. The bilateral filter mathematics is described in Equation 7.1.7.

$$I_{BF}(x,y) = \frac{1}{K} \sum_{x,y=(\hat{x},\hat{y})-N}^{(\hat{x},\hat{y})+N} \exp -\frac{||x-\hat{x}||^2 + ||y-\hat{y}||^2}{2\sigma_d^2} \exp -\frac{(I(x,y) - I(\hat{x},\hat{y}))^2}{2\sigma_r^2}, \quad (7.1.1)$$

where $I(x, y)$ is the pixel intensity in the image before applying the bilateral filter, $I_{BF}(x, y)$ is the resulting pixel intensity after applying the bilateral filter, $(\hat{x}, \hat{y})$ is the coordinates of the pixels encompassed in the bilateral filter window, K is a normalization constant given by Equation 7.1.8:

$$K = \sum_{x,y=(\hat{x},\hat{y})-N}^{(\hat{x},\hat{y})+N} \exp -\frac{||x-\hat{x}||^2 + ||y-\hat{y}||^2}{2\sigma_d^2} \exp -\frac{(I(x,y) - I(\hat{x},\hat{y}))^2}{2\sigma_r^2}. \quad (7.1.2)$$

These Equations show that the bilateral filter has three parameters. The first parameter

is the σ_d which defines the filter domain, as illustrated in equation 7.1.9, whereas equation 7.1.10 states that the filter range is defined by the second parameter denoted as σ_r. The third parameter is the bilateral filter window size $[N \times N]$.

$$\exp -\frac{||x-\hat{x}||^2 + ||y-\hat{y}||^2}{2\sigma_d^2} \tag{7.1.3}$$

$$\exp -\frac{(I(x,y) - I(\hat{x},\hat{y}))^2}{2\sigma_r^2} \tag{7.1.4}$$

The geometric spread of the bilateral filter is controlled by σ_d. As σ_d is increased, more neighbors are combined for diffusion resulting in higher smoothening, while σ_r representing the photometric spread of the bilateral. Only pixels with a percentage difference of less than σ_r are processed, while those higher than σ_r are not, according Al-Hinnawi & Daer (2015).

The following evaluates several methods of thresholding: Global thresholding, Local thresholding, including-sliding window thresholding can be in Appendix A.

Otsu (1979) considered the pixels of a given picture be represented in L gray levels $[1, 2, \ldots, L]$. The number of pixels at level i is denoted by n_i and the total number of pixels by $N = n_1 + n_2 + \cdots + n_L$. In order to simplify the discussion, the gray-level histogram is normalized and regarded as a probability distribution:

$$p_i = \frac{n_i}{N}, \qquad p_i \geq 0, \qquad \sum_{i=1}^{L} p_i = 1. \tag{7.1.5}$$

Otsu (1979) dichotomized the pixels into two classes C_0 and C_1 (background and objects, or vice versa) by a threshold at level k; C_0 denotes pixels with levels $[1, 2, \ldots, k]$, and C_1 denotes pixels with levels $[k+1, \ldots, L]$. Then the probabilities of class occurrence and the class mean levels, respectively, are given by

$$\omega_0 = Pr(C_0) = \sum_{i=1}^{k} p_i = \omega_k, \tag{7.1.6}$$

$$\omega_1 = Pr(C_1) = \sum_{i=k+1}^{L} p_i = 1 - \omega_k, \tag{7.1.7}$$

and

$$\mu_0 = \sum_{i=1}^{k} iPr(i|C_0) = \sum_{i=1}^{k} \frac{ip_i}{\omega_0} = \frac{\mu(k)}{\omega(k)}, \tag{7.1.8}$$

$$\mu_1 = \sum_{i=k+1}^{L} iPr(i|C_1) = \sum_{i=k+1}^{L} \frac{ip_i}{\omega_1} = \frac{\mu(\tau) - \mu(k)}{1 - \omega(k)}, \tag{7.1.9}$$

where

$$\omega_k = \sum_{i=1}^{k} p_i, \tag{7.1.10}$$

and

$$\mu_k = \sum_{i=1}^{k} ip_i. \tag{7.1.11}$$

The Equations 7.1.10 and 7.1.11 are the $zero^{th}$ and the first-order cumulative moments of the histogram up to the k^{th} level, respectively, and

$$\mu_\tau = \mu_L = \sum_{i=1}^{L} ip_i \tag{7.1.12}$$

is the total mean level of the original picture. For any choice of k, we have:

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_\tau, \qquad \omega_0 + \omega_1 = 1. \tag{7.1.13}$$

The class variances are given by

$$\sigma_0^2 = \sum_{i=1}^{k}(i - \mu_0)^2 Pr(i|C_0) = \sum_{i=1}^{k}(i - \mu_0)^2 \frac{p_i}{\omega_0}, \tag{7.1.14}$$

$$\sigma_1^2 = \sum_{i=k+1}^{L} (i - \mu_1)^2 Pr(i|C_1) = \sum_{i=k+1}^{L} (i - \mu_1)^2 \frac{p_i}{\omega_1}. \tag{7.1.15}$$

These require second-order cumulative moments.

Otsu also introduced the following discriminant criterion measures used in the discriminant analysis:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \qquad \kappa = \frac{\sigma_\tau^2}{\sigma_W^2}, \qquad \eta = \frac{\sigma_B^2}{\sigma_\tau^2}, \tag{7.1.16}$$

where

$$\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2 \tag{7.1.17}$$

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_\tau)^2 + \omega_1(\mu_1 - \mu_\tau)^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2. \tag{7.1.18}$$

Due to Equation 7.1.13, this becomes the function object goal (the criterion measure) of an optimization problem, and

$$\sigma_\tau^2 = \sum_{i=1}^{L}(1 - \mu_\tau)^2 p_i. \tag{7.1.19}$$

This standpoint is motivated by a conjecture that well-thresholded classes would be separated in gray levels, and conversely, a threshold giving the best separation of classes into gray levels would be the best threshold.

$$\sigma_W^2 + \sigma_B^2 = \sigma_\tau^2. \tag{7.1.20}$$

Both σ_W^2 and σ_B^2 are functions of threshold level k, but σ_τ^2 is independent of k. It is also noted that σ_W^2 is based on the second-order statistics (class variances), while σ_B^2 is based on the first-order statistics (class means). Therefore, η is the simplest measure with respect to k. Thus we adopt η as the criterion measure to evaluate the "goodness" (or separability) of the threshold at level k. The optimal threshold k^* that maximizes η, or equivalently maximizes σ_B^2 is selected in the following sequential search by using the simple cumulative quantities 7.1.10 and 7.1.11, or explicitly using the difference of

equations 7.1.6 - 7.1.9:

$$\eta(k) = \frac{\sigma_B^2}{\sigma_\tau^2} \tag{7.1.21}$$

$$\sigma^2(k) = \frac{[\mu_\tau \omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]} \tag{7.1.22}$$

and the optimal threshold k^* is:

$$\sigma_B^2(k^*) = \underset{\{1 \le k < L\}}{Max} \sigma_B^2(k) \tag{7.1.23}$$

The maximum value $\eta(k^*)$, denoted simply by η^*, can be used as a measure to evaluate the separability of classes (or ease of thresholding) for the original picture or the bimodality of the histogram.

The effective range of the gray-level histogram is

$$S^* = \{k; \omega_0\omega_1 = \omega(k)[1 - \omega(k)]\} > 0, \qquad or \qquad 0 < \omega_k < 1\}. \tag{7.1.24}$$

From the definition in 7.1.18, the criterion measure σ_B^2 (or η) takes a minimum value of zero for such k as $k \in S - S^* = \{k; \omega(k) = 0 \qquad or \qquad 1\}$ (i.e., making all pixels either C_1 or C_0, which is, of course, not our concern) and takes a positive and bounded value for $k \in S^*$. It is, therefore, obvious that the maximum always exists. All these results are classic according according to Otsu (1979).

Gupta *et al.* (2007) presented a local version of the Otsu method which looks at blocks of pixels at several resolution levels when determining the threshold. The goal is to adapt to changing backgrounds and differing font sizes. The smallest block size is adaptively chosen to be twice the dominant line height h (see Appendix for an algorithm to calculate the dominant line height automatically). This fundamental block size $2h \times 2h$ was designed so that it is large enough to contain intact letters (when located in a text region), but small enough to adapt to background changes. Several larger block sizes are also used $4h \times 4h$, $8h \times 8h$, $16h \times 16h$, $32h \times 32h$, $64h \times 64h$, and the entire image. For each block size, the image is considered to be tiled with adjacent non-overlapping blocks. This means

that a $2h \times 2h$ block is not centered in its containing $4h \times 4h$ block, but this speeds up the algorithm significantly over using centered multiresolutional blocks.

The binarization consists of the following steps (note that Steps (2) to (5) can be implemented in parallel over the $2h \times 2h$ blocks, and Step (7) is also a parallel operation).

- Step (1) - The image is completely divided up into nonoverlapping adjacent blocks of size $2h \times 2h$.
- Step (2) - For each block, an Otsu threshold is calculated based on the pixels in that block.
- Step (3) - The pixels in each block are binarized.
- Step (4) - If the ratio of binarized white pixels to binarized black pixels is less than two, then Steps $(2) to (4)$ are repeated for the next larger block which contains the given block.
- Step (5) - Each $2h \times 2h$ block is assigned the last Otsu threshold calculated for that block.
- Step (6) - Thresholds t_i for each pixel are formed by bilinear interpolation of the thresholds in Step (5).
- Step (7) - Each pixel x_i of the original image is compared to the corresponding threshold t_i to form the binarized pixel b_i .

Thresholding is a special case of pattern classification in which a one-dimensional feature space is used, the feature being the gray level of the pixel.

The threshold is a "hyper-plane" decision surface (i.e., a point) in this one-dimensional space according to Weszka (1979). If we knew the distribution of gray levels in the given ensemble of images represented here by a mixture of two Gaussian populations with given means and standard deviations then we could determine analytically the threshold that minimizes the classification error.

In the absence of such knowledge, we can approach the problem of threshold selection by performing cluster analysis on the feature space. Suppose that we construct the

gray-level histogram of the image (or ensemble); this is a plot showing how often each gray level occurs. A cluster of feature values is then nothing more than a peak on the histogram, corresponding to a densely populated range of gray levels. If we find two peaks on the histogram, it is reasonable to choose a threshold that separates these peaks, at the bottom of the valley between them, since this threshold appears to separate the gray-level population into two distinctive subpopulations.

Several methods have been proposed that produce a transformed gray-level histogram in which the valley is deeper, or is converted into a peak, and is thus easier to detect. Weszka (1979) proposed a standard approach to threshold selection for image segmentation based on locating valleys in the image's gray-level histogram.

Kapur *et al.* (1985) used a method for gray-level picture thresholding using the entropy of the histogram, maximizing the function $\phi(s) = \{H_b + H_w\}$, to obtain the maximum information between the object and background distribution in the picture.

Model-based methods have been proposed by Dawoud & Kamel (2004) and Liu & Srihari (1997).

These methods are efficient for the images in which the gray levels of the text and background pixels are separable. If the histogram of the text overlaps with that of the background, they result in improper binary images, (Valizadeh & Kabir, 2012).

Initially, a category of image was proposed where the image is divided into subimages: $\{Image(1), Image(2) \ldots, Image(M)\}$, with the levels of background noise independent and unknown. The objective of the approach is to find an optimal threshold for each subimage that would eliminate background noise, while preserving as much handwritten stroke data as possible. Suppose that threshold is given by $T(x), x \in \{1, 2, \ldots, M\}$, where M is the total number of subimages. Dawoud & Kamel (2004) showed how to optimize the binarization of a subimage (b) using information from other subimage (a).

The challenge is to show how to optimize the binarization of Image (y) using information from other another subimage, Image (x). Suppose that all subimages are binarized at a CT_i (Candidate to Threshold-i). If the Image (x) is noise free, then its gray-level statistics of the extracted pixels (pixels with gray-level lower than CT_i only) can be used to estimate the parameters of a Gaussian distribution $N_x \sim (\mu_x, \sigma_x)$. An empirical study

done by Dawoud & Kamel (2002) showed that the Gaussian distribution is the best among other distributions in representing the handwriting gray-level population. If the binarized Image (y) is noise free also, then we expect its gray-level statistics of its extracted pixels (μ_y) to be similar to N_x. When fails to eliminate Image (y) noise, these statistics will become different from N_x. Experimentally, we found this expectation to be true, regardless of the noise's pattern or characteristics. So, by testing the following null and alternative hypotheses, we can infer whether or not a eliminated background noise from Image (y):

$$H_0 : \mu_a - \mu_b = 0, \tag{7.1.25}$$

$$H_1 : \mu_a - \mu_b \neq 0. \tag{7.1.26}$$

In hypothesis testing the decision making about the state of nature is based on data. The Table 7.1 represents the true state of nature. Thus, there are two possibilities of error:

Table 7.1: Summary of Statistical Decisions.

Statistical Decision	True state of null hypothesis	
	H_0 True	H_0 False
Reject H_0	Type I error	Correct
Do not reject H_0	Correct	Type II error

1. Type I: rejecting the null hypothesis when the null hypothesis is true.
2. Type II: failing to reject the null hypothesis when the null hypothesis is false.

Where the probabilities of these errors, shown in Figure 7.1, are defined as follows:

- α = P(Type I error) = P(rejecting $H_0|H_0$ is true)
- β = P(Type II error) = P(failing to reject $H_0|H_0$ is false)
- Power = $1 - \beta$ = P(rejecting $H_0|H_0$ is false)

The hypothesis test is conducted by calculating the gray-level that corresponds to 5% of subimage (x) noise-free Gaussian distribution. A Gaussian distribution was used to model

the gray-level statistics. To corroborate that this distribution is an appropriate model, the Kolmogorov-Smirnov goodness-of-fit test was used by Dawoud & Kamel (2002).

Then, as shown in Figure 7.2, was calculated the number of pixels above this gray-level was calculated as a percentage of the total number of extracted pixels in the subimage (b). This percentage is interpreted as $\alpha(xy)$, the probability of error associated with accepting the hypothesis that the gray-level statistics of the subimage (b) obey the Gaussian model obtained from the subimage (a), where $a, b \in K$ and $K = \{1, 2, \ldots, M\}$. As long as CT_i eliminates subimage (b) background noise, this error will be small (around 5%), given that subimage (a) is noise free also. The gray-level feature for all subimages is showed in Equation 7.1.27:

$$\begin{bmatrix} - & \alpha(12) & \ldots & \alpha(1M) \\ \alpha(21) & - & \ldots & \alpha(2M) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha(M1) & \alpha(M2) & \ldots & - \end{bmatrix} \tag{7.1.27}$$

The decision to accept or reject H_0 is reached by comparing with a predetermined parameter called the gray-level rejection criterion (GRC). If error exceeds GRC, is rejected;

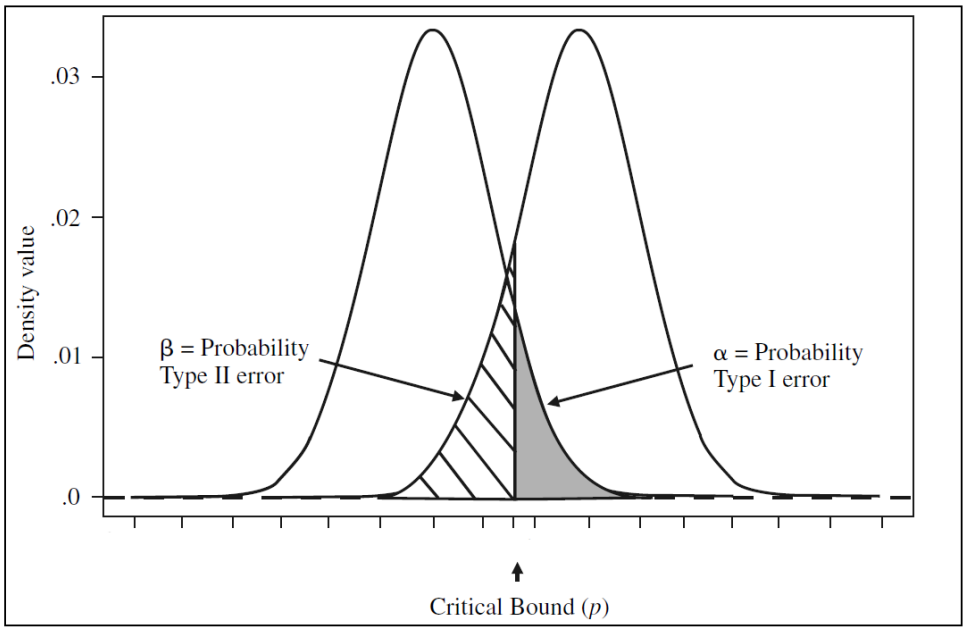


Figure 7.1: Critical bound.

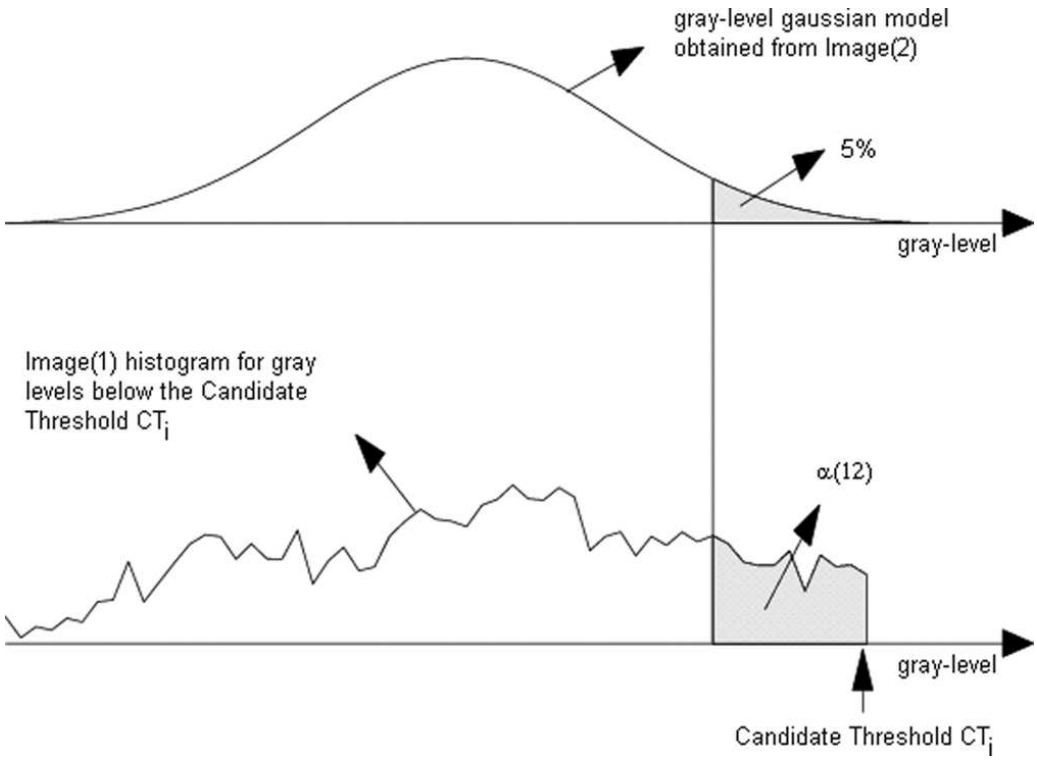


Figure 7.2: Hypothesis test (Dawoud & Kamel, 2004).

otherwise, it is accepted. When noise interferes in submage (z) at a certain CT_i, then subimage (z) can no longer be used as a valid model to evaluate other subimages. Therefore, we remove it from the list of valid subimages used to calculate at the following iterations.

When the noise interference starts at the same CT_i for two subimages: subimage (a) and subimage (b), there is a chance that neither $\alpha(yx)$ nor $\alpha(xy)$ will increase to reflect this noise interference. In this case, this feature may fail to detect the interference.

7.1.1 The IsoData Method

IsoData is one of the most popular and widely used clustering methods in geoscience applications, but it can run slowly, particularly with large data sets (Memarsadeghi, 2007). The attribute set includes properties as follows:

- Don't need to know the number of clusters.
- Algorithm splits and merges clusters.

- User defines threshold values for parameters.
- Computer runs algorithm through many iterations until threshold is reached.
- Clusters associated with fewer than the user-specified minimum number of pixels are eliminated.
- Isolated pixels are either put back in the pool for reclassification, or ignored as "unclassifiable".

Unsupervised classification techniques use clustering mechanisms to group image pixels into unlabeled classes/clusters. Later on, a human analyst assigns meaningful labels to the clusters and produces a suitably classified image.

The Isodata method works as follows:

1. Cluster centers are randomly placed and pixels are assigned based on the shortest distance to the center method.
2. The standard deviation within each cluster, and the distance between cluster centers is calculated
    - Clusters are split if one or more standard deviation is greater than the user-defined threshold.
    - Clusters are merged if the distance between them is less than the user-defined threshold.
3. A second iteration is performed with the new cluster centers.
4. Further iterations are performed until:
    - the average inter-center distance falls below the user-defined threshold,
    - the average change in the inter-center distance between iterations is less than a threshold, or
    - the maximum number of iterations is reached

7.1.2 Pun Method

Let t be the value of the threshold and define two a posteriori entropies, where H_b and H_w are:

$$H_b = -\sum_{i=0}^{t} p(i)\log_e(p(i)), \tag{7.1.28}$$

$$H_w = -\sum_{i=t+1}^{255} p(i)\log_e(p(i)), \tag{7.1.29}$$

where H_b and H_w can be regarded, respectively, as measure of the a posteriori information associated with the black and white pixels after thresholding.

Knowing the a priori entropy of the gray level histogram, Pun proposed an algorithm to determine the optimal threshold the upper bound of the posteriori entropy:

$$H = H_b + H_w. \tag{7.1.30}$$

Pun (1981)has shown the maximizing H is equivalent to maximizing the evaluation function with respect to t:

$$f(t) = \frac{H_t}{H_T}\frac{\log_e(P(t))}{\log_e[max(p_0,\ldots,p_t)]} + (1-\frac{H_t}{H_T})\frac{\log_e[1-P(t)]}{\log_e[max(p_{t+1},\ldots,p_{255})]}, \tag{7.1.31}$$

where,

$$H_t = -\sum_{i=0}^{t} p(i)\log_e(p(i)), \tag{7.1.32}$$

$$H_T = -\sum_{i=0}^{255} p(i)\log_e(p(i)), \tag{7.1.33}$$

$$P_t = \sum_{i=0}^{t} p_i. \tag{7.1.34}$$

In his second algorithm, Pun proposed the use of an anisotropic coefficient β in thresh-

olding, where,

$$\beta = \frac{H_t}{H_T} = \frac{\sum_{i=0}^{m} p_i \log_e p_i}{\sum_{i=0}^{255} p_i \log_e p_i}, \tag{7.1.35}$$

and m is the smallest integer such that:

$$\sum_{i=0}^{m} p_i \geq 0.5. \tag{7.1.36}$$

The threshold t^* is chosen such that:

$$\sum_{i=0}^{t^*} p_i = \begin{cases} 1-\beta & \text{if} \quad \beta \leq 0.5, \\ \beta & \text{if} \quad \beta > 0.5. \end{cases} \tag{7.1.37}$$

7.1.3 Kapur-Sahoo-Wong Filter

Two probability distributions (e.g., object distribution and background distribution) are derived from the original gray level distribution of the image as follows:

$$\frac{p_0}{P_t}, \frac{p_1}{P_t}, \ldots, \frac{p_t}{P_t}, \tag{7.1.38}$$

and

$$\frac{p_{t+1}}{1-P_t}, \frac{p_{t+2}}{1-P_t}, \ldots, \frac{p_{t-1}}{1-P_t}, \tag{7.1.39}$$

where t is the value of the threshold and $P_t = \sum_{i=0}^{t} p_i$. The values of the entropies H_w and H_b are calculated through Equations 7.1.40 and 7.1.41:

$$H_b = -\sum_{0}^{t} \frac{p_i}{P_t} \log_e(\frac{p_i}{P_t}), \tag{7.1.40}$$

$$H_w = -\sum_{t+1}^{t-1} \frac{p_i}{1-P_t} \log_e(\frac{p_i}{1-P_t}), \tag{7.1.41}$$

Then the optimal threshold t^* is defined as the gray level which maximizes $\phi(s) = \{H_b + H_w\}$, that is:

$$t^* = \arg\max\{H_b + H_w\}. \tag{7.1.42}$$

7.1.4 Johannsen-Bille Method

The Johannsen-Bille method choose the threshold t^* from the relation:

$$t^* = \arg\min\{S(t) + S\bar{(}t)\}, \tag{7.1.43}$$

where

The algorithm proposed by Johannsen & Bille (1982) aims at minimizing the function $S(t)$ defined as follows.

$$S(t) = \log_e(\sum_{i=0}^{t} p_i) - \frac{1}{(\sum_{i=0}^{t} p_i)}[p_t log_e p_t + (\sum_{i=0}^{t-1} p_i)\log_e(\sum_{i=0}^{t-1} p_i)], \tag{7.1.44}$$

and

$$\bar{S}(t) = \log_e(\sum_{i=t}^{t} p_{L-1}) - \frac{1}{(\sum_{i=t}^{t} p_{L-1})}[p_t log_e p_t + (\sum_{i=t+1}^{L-1} p_i)\log_e(\sum_{i=t+1}^{L-1} p_i)], \tag{7.1.45}$$

7.1.5 Yen-Chang-Chang Method

The criterion is based on the consideration of two factors. The first one is the discrepancy between the thresholded and original images and the second one is the number of bits required to represent the thresholded image. Based on a new maximum correlation criterion for bilevel thresholding, the discrepancy is defined and then a cost function that takes both factors into account is proposed for multilevel thresholding. By minimizing the cost function, the classification number that the gray-levels should be classified and the threshold value can be determined automatically. Computational analysis indicate that the number of required mathematical operations in the implementation of our algorithm

is much less that of maximum entropy criterion, according Yen (1995).

A total entropy is defined as:

$$TE(t) = E_b(t) + E_w(t) = -\log\{\sum_{i=0}^{t}[\frac{p_i}{P_t}]^2\} - \log\{\sum_{i=t+1}^{255}[\frac{p_i}{1-P_t}]^2\} \tag{7.1.46}$$

and the threshold is the argument that maximizes that expression.

The maximum entropy criterion is determined the threshold s^* such that

$$TE(s^*) = \max TE_s. \tag{7.1.47}$$

It is well-known that the thresholded image becomes more similar to the original one as the classification number increases. Hence, the discrepancy between the original and thresholded images decreases as the classification number increases. However, the total number of bits required to represent the thresholded image increases as the number of classes increases. Hence, there must exist a compromise between these two factors. Let k denote the classification number and $D(k)$ the discrepancy between the thresholded and original images. The cost function $C(.)$ that takes into account both factors is proposed as

$$C(k) = \rho[D(k)]^{\frac{1}{2}} + [\log_2(k)]^2, \tag{7.1.48}$$

where ρ is a positive weighting constant.

The first term of $C(k)$ measures the cost incurred by the discrepancy between the thresholded and original images, and the second measures the cost resulted from the number of bits used to represent the thresholded image.

The automatic thresholding criterion, according to Yen (1995), is then defined to determine the optimal classification number k^* such that

$$C(k^*) = \min C(k). \tag{7.1.49}$$

7.1.6 Otsu Threshold Method

The mean and variance of the object and background in relation to the threshold t are defined as follows.

$$\mu_b(t) = \sum_{i=0}^{t} ip_i, \tag{7.1.50}$$

$$\mu_w(t) = \sum_{i=t+1}^{255} ip_i. \tag{7.1.51}$$

The class variances are given by

$$\sigma_b^2(t) = \sum_{i=0}^{t} (i - m_b)^2 p_i, \tag{7.1.52}$$

$$\sigma_w^2 = \sum_{i=t+1}^{255} (i - m_t)^2 p_i. \tag{7.1.53}$$

The "optimal" value for this limit is the argument that maximizes the following expression

$$\eta(t) = \frac{P_t(1 - P_t)[m_b(t) - m_w(t)]^2}{P_t\sigma_b^2(t) + (1 - P_t)\sigma_w^2(t)}. \tag{7.1.54}$$

7.1.7 Mello-Lins Algorithm

The algorithm by Mello & Lins (2002) and Mello & Lins (2000) looks for the most frequent gray level of the image and takes it like initial threshold to evaluate the values H_b, H_w and H by equations 7.1.28, 7.1.29 and 7.1.30. Based on the value of H, three classes of documents were identified, according to Mello (2006). It was defined two multiplicative factors and the entropy H determines the value of weights m_b and m_w, as follows:

- If $H \leq 0.25$, (documents with few parts of text or vary faded ink), then $m_w = 2$ and $m_b = 3$.
- If $0.25 < H < 0.30$, (the most common cases), then $m_w = 1$ and $m_b = 2.6$.

- If $H \geq 0.30$, (documents with many blacks areas), then $m_w = 1$ and $m_b = 1$,

then, the threshold is directly calculated by:

$$t = (m_b H_b + m_w H_w). \tag{7.1.55}$$

7.1.8 Silva-Lins-Rocha Approach

Thus, the entropy of the a priori source is given by

$$H = -\sum_{i=0}^{255} p_i \log_2 p_i, \tag{7.1.56}$$

where p_i is provided by Equation 7.1.5. As the resulting image is binarized, the distribution of its histogram may be seen as a distribution of a binary source (a posteriori source). The entropy of the a posteriori source is given by:

$$H'(t) = h(P_t), \tag{7.1.57}$$

where $h(p) = -p \log_2(p) - (1-p) \log_2(1-p)$ is the binary entropy function, according to Abramson (1963), and P_t is provided by Equation 7.1.6. Next, one makes an extension of a binary source to represent without losses all the 256 symbols of the a priori source. This new binary source is called the a priori binary source. The value of the entropy of this new source is given by

$$H_{\textbf{(a priori binary source)}} = \frac{H}{\log_2(256)} = \frac{H}{8}. \tag{7.1.58}$$

One then looks for a value of t such that the entropy of the a posteriori source is as close as possible to the value of the entropy of the a priori binary source, i.e., one looks for the following equality:

$$H'(t) = H_{\textbf{(a priori binary source)}}. \tag{7.1.59}$$

This argument maps the distribution of the a posteriori source onto the distribution of the a priori binary source.

Applying Equations 7.1.57 and 7.1.58 to 7.1.59 it follows that

$$h(P_t) = \frac{H}{8}. \tag{7.1.60}$$

The behavior of the binary entropy function, Figure 7.3, must be taken into account as follows. It must be taken into account that the target images are of documents, with a much higher frequency of background (paper) pixels than object (print or written) ones. Thus, it is reasonable to work with the argument of P_t within the interval $[0.0, 0.5]$. In this interval, the entropy function is injective, thus there is only one value of P_t that satisfies the equation, unless p_i is zero. In such a case it would not matter if the calculated limit were i or $i-1$. The target of the proposed algorithm is to filter out the back-to-front interference in binarization. Due to its features, the interference raises the value of the a priori source's entropy. A loss factor $\gamma H_{\textbf{(a priori binary source)}}$, according to Abramson (1963), experimentally determined, is introduced to reduce the presence of the interference.

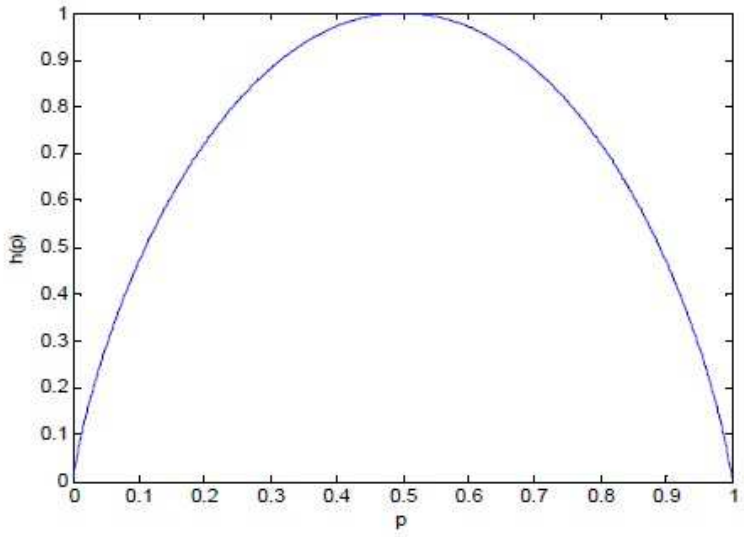


Figure 7.3: Behavior of the binary entropy function h(p).

Thus, the following relation holds:

$$H'(t) = \gamma[H_{\textbf{(a priori binary source)}}][H_{\textbf{(a priori binary source)}}] \tag{7.1.61}$$

Once the bases of the algorithm are presented, its steps are now detailed.

1. Calculate H, the entropy of the image histogram.

2. Scan the t levels, calculating of each of them the distributions $(P_t, 1 - P_t)$, while $P_t \leq 0.5$, and the entropy associated with that distribution $H'(t) = h(P_t)$;

3. Determine the "optimal" limit that minimizes $|e(t)|$ given as:

$$e(t) = |\frac{H'(t)}{\frac{H}{8}} - \gamma(H/8)|. \tag{7.1.62}$$

7.1.9 Wu-Lu Algorithm

According to Kapur-Sahoo-Wong method, the optimal threshold t^*, as defined in Equation 7.1.42 is defined as the gray level which maximizes $\phi(s) = \{H_b + H_w\}$.

As a result, a gray level minimizing the difference between the entropy of the object and that of the background $\phi^{'}(s) = \{H_b - H_w\}$ will be the desired threshold, according Equation 7.1.63, Wu (1998).

$$t^* = \arg\min\{H_b - H_w\}. \tag{7.1.63}$$

8 APPENDIX B

The binarization methods reported in the literature are generally global or local. Global methods find a single threshold value by using some criteria based on the gray levels of the image. These methods compare the gray level of each pixel with this threshold and classify it as a text or background pixel. Global thresholding based on clustering Otsu (1979), entropy minimization Kapur *et al.* (1985), and valley seeking in the intensity histogram Weszka (1979) as well as feature-based and model-based Liu & Srihari (1997) methods has been proposed.

8.1 Annotated Bibliography

8.1.1 Global Thresholding

1. Based on Statistical Analysis

   (a) Nafchi *et al.* (2014) - Phase-based binarization of ancient document images: Model and applications. IEEE Transactions on Image Processing, p. 1-14, 2014.

   Was introduced an image binarization method that uses the phase information of the input image, and robust phase-based features extracted from that image are used to build a model for the binarization of ancient manuscripts. Phase-preserving denoising followed by morphological operations are used to preprocess the input image. Then, two phase congruency features, the maximum moment of phase congruency covariance and the locally weighted mean phase angle, are used to perform the main binarization.

   (b) Brocher (2014) - Qualitative and Quantitative Evaluation of Two New Histogram Limiting Binarization Algorithms. International Journal of Image Processing, v. 8, n. 2, p. 30-48, 2014.

   Quantitative measure for relative binarization quality assessment for individual images. Two algorithms based on statistical histogram values and initial

histogram limitation are considered. This mode-limited mean (MoLiM) as well as the differential-limited mean (DiLiM) algorithms were implemented in ImageJ and compared to 22 existing global as well as local automatic binarization algorithms using the evaluation method described here.

(c) Ramirez-Ortegon *et al.* (2014) - An analysis of the transition proportion for binarization in handwritten historical documents. Pattern Recognition, v. 47, p. 2635-2651, 2014.

A mathematical analysis of the transition proportion for the normal threshold (NorT) based on the transition method. The analysis extends to thresholding methods that rely on Bayes rule, and it also gives the mathematical bases for potential applications of the transition proportion as a feature to estimate stroke width and detect regions of interest. In the majority of our experiments,we used a database composed of small images that were extracted from DIBCO 2009 and H-DIBCO 2010 benchmarks.

(d) Moghaddam & Cheriet (2012) - AdOtsu: An adaptive and parameterless generalization of Otsu's method for document image binarization. Pattern Recognition, v. 45, p. 2419-2431, 2012.

An adaptive and parameterless generalization of Otsu's method is presented. The adaptiveness is obtained by combining grid-based modeling and the estimated background map. The parameterless behavior is achieved by automatically estimating the document parameters, such as the average stroke width and the average line height.

(e) Monte da Silva *et al.* (2008) - A New and Efficient Algorithm to Binarize Document Images Removing Back-to-Front Interference. Journal of Universal Computer Science, v. 14, n. 2, p. 293-313, 2008.

Presented a fast entropy-based segmentation method for generating high-quality binarized images of documents with back-to-front interference.

(f) Gupta *et al.* (2007) - OCR binarization and image pre-processing for searching historical documents. The Journal of the Pattern Recognition Society, v. 40, p. 389-397, 2007.

Was considered the problem of document binarization as a pre-processing step for optical character recognition (OCR) for the purpose of keyword search of historical printed documents. A number of promising techniques from the literature for binarization, pre-filtering, and post-binarization denoising were implemented along with newly developed methods for binarization: an error diffusion binarization, a multiresolutional version of Otsu's binarization, and denoising by despeckling.

(g) Gatos *et al.* (2006) - Adaptive degraded document image binarization. The Journal of the Pattern Recognition Society, v. 39, p. 317-327, 2006.

Was followed several distinct steps: a pre-processing procedure using a low-pass Wiener filter, a rough estimation of foreground regions, a background surface calculation by interpolating neighboring background intensities, a thresholding by combining the calculated background surface with the original image while incorporating image up-sampling and finally a post-processing step in order to improve the quality of text regions and preserve stroke connectivity.

(h) Liu & Srihari (1997) - Document Image Binarization Based on Texture Features. IEEE Transaction on Pattern Analysis and Machine Inteligence, v. 19, n. 5, p. 540-544, 1997.

This algorithm consists of three steps: 1) Candidate thresholds are produced through iterative use of Otsu's algorithm; 2) Texture features associated with each candidate threshold are extracted from the run-length histogram of the accordingly binarized image; 3) The optimal threshold is selected so that desirable document texture features are preserved.

(i) Kittler & Illingworth (1986) - Minimum Error Thresholding. Pattern Recognition, v. 19, n. 1, p. 41-47, 1986.

A computationally efficient solution to the problem of minimum error thresholding is derived under the assumption of object and pixel grey level values being normally distributed. The method is applicable in multithreshold selection.

(j) Otsu (1979) - A Thresholding Selection Method from Gray-Level Histogram.

IEEE Transaction on Systems, Man and Cybernetics, v. SMC-9, n. 1, p. 62-66, 1979.

(Otsu, 1979) dichotomizes the pixels into two classes C_0 and C_1 (background and objects, or vice versa) by a threshold at level k; C_0 denotes pixels with levels $[1, 2, \ldots, k]$, and C_1 denotes pixels with levels $[k+1, \ldots, L]$.

(k) Weszka (1979) - Histogram Modification for Threshold Selection. IEEE Transaction on Systems, Man and Cybernetics, v. SMC-9, n. 1, p. 38-52, 1979.

The transformed histograms used in several methods can be obtained by creating (gray level, edge value) scatter plots, and computing various weighted projections of these plots on the gray-level axis. Using this unified approach makes it easier to understand how the methods work and to predict when a particular method is likely to be effective.

2. Based on Entropy

(a) Kapur *et al.* (1985) - A new method for gray-level picture thresholding using the entropy of the histogram. Computer Vision Graphics and Image Processing, v. 29, p. 273-285, 1985.

This article was propused a algorithm based on the entropy concept. Using the entropy of histogram are calculated of picture thresholding.

8.1.2 Local Thresholding

1. Based on Statistical Analysis

(a) Ntirogiannis *et al.* (2014a) - A combined approach for the binarization of handwritten document images. Pattern Recognition Letters, v. 35, p. 3-15, 2014.

A combination of a global and a local adaptive binarization method at connected component level is proposed that aims in an improved overall performance. Initially, background estimation is applied along with image normalization based on background compensation. Afterwards, global binarization is

performed on the normalized image. In the binarized image very small components are discarded and representative characteristics of a document image such as the stroke width and the contrast are computed. Furthermore, local adaptive binarization is performed on the normalized image taking into account the aforementioned characteristics. Finally, the two binarization outputs are combined at connected component level.

(b) Arruda & Mello (2014) - Binarization of Degraded Document Images Based on Combination of Contrast Images. 14th International Conference on Frontiers in Handwriting Recognition, p. 615-620, 2014.

The binarization of grayscale images of degraded documents is based on the creation of three different structural contrast images, the representation of these features in a 2D feature space that is further partitioned. A weak bi-level image and a strong bi-level image are created and combined to produce the final image.

(c) Shah *et al.* (2014) - Handwritten Character Recognition using Radial Histogram. International Journal of Research in Advent Technology, v. 2, n. 4, p. 24-28, 2014.

The authors concentrated on structural characteristics for feature extraction. For this we have approached a new methodology of radial histogram. Radial histogram is computed from 72 directions at 5 degree interval. Number of black pixels are counted for each of 72 directions and hence getting total 72 unique feature vectors. After getting 72 feature vectors using Euclidean distance classifier the characters are classified accordingly.

(d) Rabeux *et al.* (2013) - Quality evaluation of ancient digitized documents for binarization prediction. 12th International Conference on Document Analysis and Recognition, p. 113-117, 2013.

Was intended to characterize the degradation of a document image by using different features based on the intensity, quantity and location of the degradation. These features allow us to build prediction models of binarization algorithms that are very accurate according to R^2 values and p-values. The prediction

models are used to select the best binarization algorithm for a given document image. Obviously, this image-by-image strategy improves the binarization of the entire dataset.

(e) Elzobi *et al.* (2013) - A Hidden Markov Model-based Approach with an Adaptive Threshold Model for Off-line Arabic Handwriting Recognition. 12th International Conference on Document Analysis and Recognition, p. 945-949, 2013.

An Hidden Markov Model was proposed - based approach that built upon an explicit segmentation module. And shape representative based rather than sliding window based features, are extracted and used to build a reference as well as a confirmation model for each letter in each handwritten form. Additionally, we constructed an HMM-based threshold model by ergodically connecting all letter models, in order to detect false segmentation as well as nonletter segments.

(f) Alaei *et al.* (2013) - Logo Detection Using Painting Based Representation and Probability Features. 12th International Conference on Document Analysis and Recognition, p. 1267-1271, 2013.

A coarse-to-fine logo detection scheme for document images is proposed. At the coarse level of the proposed scheme, content of a document image is pruned utilizing a decision tree and a small number of features such as frequency probability (FP), Gaussian probability (GP), height, width, and average density computed for patches. The patches are extracted employing the piece-wise painting algorithm (PPA) used for text-line segmentation. The fine level of the proposed scheme refines the detection results by integrating shape context descriptors and a Nearest Neighbor (NN) classifier.

(g) Shaikh *et al.* (2013) - A new image binarization method using iterative partitioning. Machine Vision and Applications, v. 24, p. 337âĂŞ350, 2013.

The proposed method is an iterative partitioned-based approach. The image that has to be thresholded is logically partitioned into four rectangular parts. This method utilizes the concept of minimizing within class variance Otsu

(1979) for calculating threshold in each partition that have two sharp peaks in the gray scale histogram.

(h) Gaceb *et al.* (2013) Adaptative Smart-Binarization Method. 12th International Conference on Document Analysis and Recognition, p. 118-122, 2013.

The quality of each pixel is estimated using a hierarchical local thresholding in order to classify it as foreground, background or ambiguous pixel. The ambiguous pixels that represent the degraded zones cannot be binarized with the same local thresholding. The global quality of the image is thus estimated from the density of theses degraded pixels. If it is considered as degraded, the authors apply a second separation on the ambiguous pixels to separate them into background or foreground.

(i) Shi *et al.* (2012) - Shape based local thresholding for binarization of document images. Pattern Recognition Letters, v. 33, p. 24-32, 2012.

Stroke width of handwritten and printed characters in documents is utilized as the shape feature. As a result, in addition to the intensity analysis, the proposed algorithm introduces the stroke width as shape information into local thresholding.

(j) Sumathi *et al.* (2012) - A Survey on Various Approaches of Text. International Journal of Computer Science and Engineering Survey, v. 3, n. 4, p. 27-42, 2012.

The methods were based on morphological operators, wavelet transform, artificial neural network, skeletonisation operation, edge detection algorithm, histogram technique etc. All these techniques have their benefits and restrictions. This article discusses various schemes proposed earlier for extracting the text from an image.

(k) Valizadeh & Kabir (2012) - Binarization of degraded document image based on feature space partitioning and classification. IJDAR, v. 15, p. 57-69, 2012.

These regions are labeled as text or background using the result of a basic binarization algorithm applied on the original image. This algorithm consists

of feature extraction, feature space partitioning, partition classification, and finally pixel classification stages, where each pixel of the image is classified as either text or background based on the label of its corresponding region in the feature space. This algorithm splits the feature space into text and background regions without using any training dataset.

(l) Henault *et al.* (2012) - A local linear level set method for the binarization of degraded historical document images. International Journal on Document Analysis and Recognition, v. 15, p. 101-124, 2012.

Binarization is accomplished by taking advantage of local probabilistic models and of a flexible active contour scheme. More specifically, local linear models are used to estimate both the expected stroke and the background pixel intensities. This information is then used as the main driving force in the propagation of an active contour.

(m) Bataineh *et al.* (2011) - An adaptive local binarization method for document images based on a novel thresholding method and dynamic windows. Pattern Recognition Letters, v. 32, p. 1805-1813, 2011.

A local binarization method based on a simple, novel thresholding method with dynamic and flexible windows was proposed. The method adopted the local approach and it is used two elements: a dynamic method to divide the image into the windows based on the characteristics of the image and a new method that can determine the appropriate value of the thresholding of each window.

(n) Hedjam *et al.* (2011) - A spatially adaptive statistical method for the binarization of historical manuscripts and degraded document images. Pattern Recognition, v. 44, n. 9, p. 2184-2196, 2011.

This approach is an adaptive method for the binarization of historical manuscripts and degraded document images. The proposed approach is based on maximum likelihood (ML) classification and uses a priori information and the spatial relationship on the image domain. In contrast with many conventional methods that use a decision based on thresholding, the proposed method performs a soft decision based on a probabilistic model. The main idea is that, from an

initialization map (under - binarization) containing only the darkest part of the text, the method is able to recover them a in text in the document image, including low - intensity and weak strokes. To do so, fast and robust local estimation of text and background features is obtained using grid-based modeling and in painting techniques; then, the ML classification is performed to classify pixels into black and white classes.

(o) Lazaro *et al.* (2010) - Neuro semantic thresholding using OCR software for high precision OCR applications. Image and Vision Computing, v. 28, p. 571-578, 2010.

It is presented a way of obtaining a threshold that depends both on the image and the final application using a semantic description of the histogram and a neural network. The input image histogram is smoothed and its derivative is found. Using a polygonal version of the derivative and the smoothed histogram, a new description of the histogram is calculated. Using this description and a training set, a general neural network is capable of obtaining an optimum threshold for our application.

(p) Mukherjee & Kanrar (2010) - Enhancement of Image Resolution by Binarization. International Journal of Computer Applications, v. 10, n. 10, p. 15-19, 2010.

Two novel algorithms to determine threshold values for the pixels value of the gray scale image are presented. The performance estimation of the algorithm utilizes test images with, the evaluation metrics for Binarization of textual and synthetic images.

(q) Pai *et al.* (2010) - Adaptive thresholding algorithm: Efficient computation technique based on intelligent block detection for degraded document images. Patern Recognition, v. 43, p. 3177-3187, 2010.

Integrating the advantages of global and local methods allows the proposed algorithm to divide the document image into several regions. A threshold surface is then constructed based on the diversity and the intensity of each region to derive the binary image.

(r) Gupta *et al.* (2007) - OCR binarization and image pre-processing for searching historical documents. Computer Vision Graphics and Image Processing, v. 29, p. 273-285, 1985.

This is a local version of the Otsu method which considers blocks of pixels at several resolution levels when determining the threshold. The goal is to adapt to changing backgrounds and differing font sizes.

(s) Cavalcanti *et al.* (2006) - A Heuristic Binarization Algorithm for Documents with Complex. 13th IEEE International Conference on Image Processing, p. 389-392, 2006.

The proposed approach performs binarization by using a heuristic algorithm with two different thresholds and the combination of the thresholded images. The method is suitable for binarization of complex background document images. In experiments, it obtained better results than classical techniques in the binarization of real bank checks.

(t) Dawoud & Kamel (2004) - Iterative Multimodel Subimage Binarization for Handwritten Character Segmentation. IEEE Transactions on Image Processing, v. 13, n. 9, p. 1223-1230, 2004.

Was considered the image as the collection of subimages. Each subimage provides a statistical model for the handwritten characters that can be used to optimize the binarization of other subimages based on gray-level and stroke-run features. The proposed method uses these multimodels to iteratively arrive at the optimal threshold for each subimage. It can be applied to different types of documents where prior knowledge about the noisiness of the subimages is not available.

(u) Dawoud & Kamel (2002) - Iterative model-based binarization algorithm for cheque images. International Journal on Document Analysis and Recognition, v. 5, p. 28-38, 2002.

The main contribution of this approach is optimizing the binarization of a part of the document image that suffers from noise interference, referred to as the Target Sub-Image (TSI), using information easily extracted from another

noise-free part of the same image, referred to as the Model Sub-Image (MSI). Simple spatial features extracted from MSI are used as a model for handwriting strokes.

(v) Sauvola & Pietikainem (2000) - Adaptive document image binarization. The Journal of the Pattern Recognition Society, v. 33, p. 225-236, 2000.

Two new algorithms are applied to determine a local threshold for each pixel. The performance evaluation of the algorithm utilizes test images with ground-truth, evaluation metrics for binarization of textual and synthetic images, and a weight-based ranking procedure for the final result presentation.

2. Based on Energy Function

(a) Milyaev *et al.* (2013) - Image binarization for end-to-end text understanding in natural images. 12th International Conference on Document Analysis and Recognition, p. 128-132, 2013.

This papers evaluates the performance of several well-known image binarization methods on established ICDAR benchmarks across different metrics, including segmentation accuracy and the final word recognition accuracy demonstrated by an OCR engine applied to the binarization result. As a result of this comparison we select the top methods and compare them within the most interesting end-to-end text detection and recognition scenario. A standard binarization method such as non-linear Niblack (1986) in combination with an off-the-shelf OCR module shows performance competitive to fancier state-of-the-art text understanding methods.

(b) Howe (2013) - Document binarization with automatic parameter tuning. International Journal on Document Analysis and Recognition, v. 16, n. 3, p. 247-258, 2013.

A promising approach to binarization based upon simple principles was examined, and showed that its success depends most significantly upon the values of two key parameters. It further describes an automatic technique for setting these parameters in a manner that tunes them to the individual image, yielding

a final binarization algorithm that can cut total error by one third with respect to the baseline version.

(c) Su *et al.* (2013) - Robust Document Image Binarization Technique for Degraded Document Images. IEEE Transactions on Image Processing, v. 22, n. 4, p. 1408-1417, 2013.

Given a degraded document image, an adaptive contrast map is first constructed and the text stroke edges are then detected through the combination of the binarized adaptive contrast map and the canny edge map. The text is then segmented based on the local threshold that is estimated from the detected text stroke edge pixels. Some post-processing is further applied to improve the document binarization quality.

3. Others

(a) Diamantatos *et al.* (2014) - Binarization: a Tool for Text Localization. 14th International Conference on Frontiers in Handwriting Recognition, p. 649-654, 2014.

A procedure of binarization is applied in order to create appropriate images for the text detection. The connected components of the image are extracted and some heuristic rules are applied in order to identify areas containing text. Finally, the overlaps are handled and the false detections are rejected.

(b) Parkera *et al.* (2014) - Robust Binarization of Degraded Document Images Using Heuristics. Document Recognition and Retrieval XXI - SPIE-IS and T Electronic Imaging, v. 9021, p. 1-12, 2014.

Principal component analysis (PCA) was used to convert the input color image (which can be viewed as collection of 3-dimensional RGB values) to a greyscale image. Identifying locally dark pixels is the first of two steps using the greyscale image as input. Each pixel in the greyscale image is analyzed separately by applying Otsu's method to a snippet of local pixels extracted from the greyscale image. A pixel is added to the set of "locally dark" pixels if the pixel is set to black when Otsu's method is applied to its corresponding snippet of local

pixels. If a pixel is to be black in the final output image it must be flagged as "locally dark" in this step. This requirement means that the output of this step can be viewed as a filter each pixel must pass to be included in the final output.

(c) Kumar & Bhatia (2014) - A Detailed Review of Feature Extraction in Image Processing Systems. Fourth International Conference on Advanced Computing and Communication Technologies, p. 5-12, 2014.

We are going to discuss various types of features, feature extraction techniques and explaining in what scenario, which features extraction technique, will be better.

(d) Chen *et al.* (2013) - Hybrid Page Segmentation with Efficient Whitespace Rectangles Extraction and Grouping. 12th International Conference on Document Analysis and Recognition, p. 958-962, 2013.

The method extracts whitespace rectangles based on connected component analysis, and filters whitespace rectangles progressively incorporating foreground and background information such that the remaining rectangles are likely to form column separators.

(e) Seki *et al.* (2013) - Color Drop-out Binarization Method for Document Images with Color Shift. 12th International Conference on Document Analysis and Recognition, p. 123-127, 2013.

The method is based on the following three calculation steps. First, line and character areas are estimated coarsely by using form structure analysis and subtracting background from images; second, the color shift is removed by using morphological processing; third, each pixel of the background subtracted images is discriminated into character strings and lines precisely by dynamic color classification.

(f) Lins *et al.* (2011) - HistDoc v. 2.0 Enhancing a Platform to Process Historical Documents. Historical Document Imaging and Processing, September, p. 169-176, 2011.

The HistDoc platform was developed as a plug-in in ImageJ. Is able to work with monochromatic documents removing framing noisy borders, correcting orientation and skew, and removing salt-and-pepper noise in images.

(g) Sokratis *et al.* (2011) - Learning Structure and Schemas from Documents. Springer-Verlag Berlin Heidelberg, 2011. Cap. A Hybrid Binarization Technique for Document Images, p. 165-179.

A hybrid approach is presented that first applies a global thresholding technique and, then, identifies the image areas that are more likely to still contain noise. Each of these areas is re-processed separately to achieve better quality of binarization.

(h) Lu *et al.* (2010) - Document image binarization using background estimation and stroke edges. International Journal on Document Analysis and Recognition, v. 13, p. 303-314, 2010.

The technique is based on the observations that the text documents usually have a document background of the uniform color and texture and the document text within it has a different intensity level compared with the surrounding document background. The proposed technique first estimates a document background surface through an iterative polynomial smoothing procedure. Different types of document degradation are then compensated by using the estimated document background surface. Finally, the document text is segmented by a local threshold that is estimated based on the detected text stroke edges.

(i) Trier & Jain (1995) - Goal-Directed Evaluation of Binarization Methods.IEEE Transaction on Pattern Analysis and Machine Intelligence, v. 17, n. 12, p. 1191-1201, 1995.

The evaluation of binarization methods is in the context of digit recognition, so we define the performance of the character recognition module as the objective measure.

(j) Niblack (1986) - An Introduction to Digital Image Processing. Prentice Hall, 1986.

The algorithm calculates a pixelwise threshold by sliding a rectangular window over the gray level image. The computation of threshold is based on the local mean μ and the standard deviation σ of all the pixels in the window.

8.1.3 Dithering

1. Dagar *et al.* (2012) - High performance Computing Algorithm Applied in Floyd Steinberg Dithering. International Journal of Computer Applications, v. 43, n. 23, p. 11-13, 2012.

   A parallel Floyd Steinberg Dithering algorithm for multi core architecture was implemented. The algorithm is based on error dispersion. This algorithm is commonly used by image manipulation software, for example when an image is converted into GIF format.

2. Omohundro (1990) - Floyd-Steinberg Dithering. International Computer Science Institute, October, 1990.

   Omohundro (1990) presented an efficient implementation technique for dithering by error diffusion. An algorithm was suggested by Floyd & Steinberg (1976). The idea of the so-called "error diffusion" approach to dithering is to produce a pattern of pixels such that the average intensity over regions in the output bitmap is approximately the same as the average over the same region in the original image. The idea is to keep track of the error made in replacing each original pixel by a single bit and to add this error to the intensity of neighboring pixels in such a way that the average remains close to the original.

8.1.4 Competition on Document Image Binarization

1. Ntirogiannis *et al.* (2014b) - ICFHR2014 Competition on Handwritten Document Image Binarization. 14th International Conference on Frontiers in Handwriting Recognition, 809-813, 2014b.

   The best method was Arruda & Mello (2014) - Binarization of Degraded Document Images Based on Combination of Contrast Images. 14th International Conference

on Frontiers in Handwriting Recognition, p. 615-620, 2014.

2. Diem *et al.* (2013) - ICDAR2013 Competition on Handwritten Digit Recognition (HDRC 2013). 12th International Conference on Document Analysis and Recognition, p. 1454-1459, 2013.

   The best method was Codrescu, C. - Temporal Processing Applied to Speech Recognition, in Intelligent Systems Design and Applications, 2004, and Badea, C. - Chapter 3 - FIR Neuron Modeling, in Approximate Dynamic Programming for Real-Time Control and Neural Modeling, F. Ionescu and D. Stefanoiu, Eds. Steinbeis Edition, 2004.

3. Pratikakis *et al.* (2013) - ICDAR 2013 Document Image Binarization Contest (DIBCO 2013). 12th International Conference on Document Analysis and Recognition, p. 1471-1476, 2013.

   Submitted by Su *et al.* (2013) - Robust Document Image Binarization Technique for Degraded Document Images. IEEE Transactions on Image Processing, v. 22, n. 4, p. 1408-1417, 2013.

   This research group has submitted two algorithms. The best method was from University of Bern, Switzerland (Anguelos Nicolaou): This method is a slightly modified version of a previous submission in H-DIBCO 2010 [3]. The method uses a local version of Otsu (1979). Using the integral image of histograms, we can obtain the histogram of any rectangular region in constant complexity. For each pixel, the Otsu threshold of a square window with the size of the minimum dimension of the grayscale image is attributed. As long as it is classified as "foreground" the window is reduced by a factor of 11/20 up to 6 times. Then all connected components become filtered out with a size less than 25 pixels assuming they are salt and pepper noise.

4. Papandreou *et al.* (2013) - ICDAR2013 Document Image Skew Estimation Contest (DISEC'13). 12th International Conference on Document Analysis and Recognition, p. 1476-1480, 2013.

   The best method was Koo, H. I. and Cho, N. I. Skew estimation of natural images based on a salient line detector, Journal of Electronic Imaging, vol. 22, no. 1, 2013.

5. Antonacopoulos *et al.* (2013) - ICDAR2013 Competition on Historical Newspaper Layout Analysis (HNLA2013). 12th International Conference on Document Analysis and Recognition, p. 1454-1458, 2013.

   The best method was Chen *et al.* (2013) for OCR Scenario - Hybrid Page Segmentation with Efficient Whitespace Rectangles Extraction and Grouping. 12th International Conference on Document Analysis and Recognition, p. 958-962, 2013.

6. Pratikakis *et al.* (2010) - H-DIBCO 2010 - Handwritten Document Image Binarization Competition. 12th International Conference on Frontiers in Handwriting Recognition, p. 727-732, 2010.

   The best method was Su *et al.* (2013) Robust Document Image Binarization Technique for Degraded Document Images. IEEE Transactions on Image Processing, v. 22, n. 4, p. 1408-1417, 2013.

7. Pratikakis *et al.* (2011) - ICDAR 2011 Document Image Binarization Contest (DIBCO 2011). 2011 International Conference on Document Analysis and Recognition, p. 1506-1510, 2011.

   The best method was Lelore & Bouchara (2011) - Super-resolved binarization of text based on the FAIR algorithm. 2011 International Conference on Document Analysis and Recognition, 839-843, 2011.

8. Gatos *et al.* (2009) - ICDAR 2009 Document Image Binarization Contest (DIBCO 2009). 10th International Conference on Document Analysis and Recognition, p. 1375-1382, 2009.

   The best method was Lu *et al.* (2010) - Document image binarization using background estimation and stroke edges. International Journal on Document Analysis and Recognition, v. 13, p. 303-314, 2010.

9 APPENDIX C

9.1 Program code in C^{++}

```
#include <C:\opencv300\build\include\opencv\cv.h>
#include <C:\opencv300\build\include\opencv\highgui.h>
#include <iostream>
#include <stdio.h>
#include <C:\opencv300\build\include\opencv2\opencv.hpp>
#include <C:\opencv300\build\include\opencv\compat.hpp>
#include <C:\opencv300\build\include\opencv2\objdetect\objdetect.hpp>
#include <C:\opencv300\build\include\opencv2\highgui.hpp>
#include <C:\opencv300\build\include\opencv2\imgproc\imgproc.hpp>
#include <sstream>
#include <stdlib.h>
#include <C:\opencv300\build\include\opencv\ml.h>
#include <math.h>
#include <string>
#include <fstream>
using namespace std;
using namespace cv;
using namespace ml;
void main()
{
	int kernel = 0;
	int kernel1 = 0;
	int kernel2 = 0;
	Mat src3 = imread("IMF.jpg", CV_LOAD_IMAGE_UNCHANGED);//carregando imagem sem ruido
(jpg RGB) e salvando na variável src3
	Mat src4;
	Size size(src3.cols * 1, src3.rows * 1);//tamanho do redimensionando da imagem sem ruído
	resize(src3, src4, size);//redimensionamento da imagem src3 e salvando na variável src4
	Mat src1 = imread("IMV.jpg", CV_LOAD_IMAGE_UNCHANGED);//carregando "imagem ruido" e
salvando na variável src1
	Mat src;
	Size size1(src1.cols * 1, src1.rows * 1);//tamanho do redimensionando da "imagem ruído "
	resize(src1, src, size1);//redimensionamento da imagem src1 e salvando na variável src
	Mat aux3 = imread("IMF.jpg", CV_LOAD_IMAGE_UNCHANGED);//variavel aux3 guarda imagem
original para comparação posteriormente
	Mat escalagrayoriginal = imread("IMF.jpg", CV_LOAD_IMAGE_GRAYSCALE);
	IplImage* src42 = cvCloneImage(&(IplImage)escalagrayoriginal);//converte a imagem original do
tipo Mat para o tipo IplImage, salva em src42
	imshow("Frente", src4);//mostra a imagem original sem ruído
	imshow("Verso", src);//mostra a "imagem ruído"
	waitKey(0);//espera key para continuar o programa
	stringstream ss;
	string folderNameCodCompleto = "Imagens_Codigo_Completo";
	string folderCreateCommandCodCompleto = "mkdir " + folderNameCodCompleto;
	system(folderCreateCommandCodCompleto.c_str());
	string typeCOD = ".jpg";
	imwrite("Imagens_Codigo_Completo/Imagem_Frente.jpg", src4);
	imwrite("Imagens_Codigo_Completo/Imagem_Verso.jpg", src); // antes do filtro guassiano
	Mat dst9; //variável para armazenar a saída do filtro gaussiano
	char zBuffer9[35]; //variável para armazenar caracteres
```

```
	for (int i = 1; i < 3; i = i + 2)
	{
		_snprintf_s(zBuffer9, 35, "Kernel Size : %d x %d", i, i);//copia do texto "Kernel Size ..." para a variável ZBuffer9
		// bilateralFilter(src, dst9, i, i * 4, i / 4);//filtro bilateral(imagem de entrada,imagem de saída(mesmo tipo da imagem de entrada.Ex.: Mat,Iplimage...),diametro de cada vizinhança do pixel,Desvio padrão no espaço de cor,Desvio padrao no espaço de coordenadas(em termo de pixel))
		GaussianBlur(src, dst9, Size(i, i), 0, 0);//filtro Gaussiano(imagem de entrada,imagem de saída(mesmo tipo da imagem de entrada.Ex.: Mat,Iplimage...),Kernel Size, Gaussian kernel standard deviation in X direction,Gaussian kernel standard deviation in Y direction)
		putText(dst9, zBuffer9, Point(src4.cols / 128, src4.rows / 32), FONT_HERSHEY_SIMPLEX, 0.4, Scalar(255, 159, 128), 1);//colocar o texto em uma imagem putText(imagem que irá colocar o texto, texto que irá colocar tipo char ou string, coordenada onde colocará o texto, Fonte,Tamanho da Fonte, Cor da fonte,tipo da linha(variável inteiro))
		imshow("Gaussian Blur Smoothed Image", dst9);//mostrar a saída após o filtro em cada iteração
		int c = waitKey(2000); //espera dois segundos para proxima iteração
		kernel = i;
		if (c == 27)//if the "esc" key is pressed during the wait, return
		{
			return;
		}
	}
	Vec3b pixel, pixel1;//variáveis para capturas das tonalidades Red, Green e Blue
	float alfa = 1;//Fator de ruído (0 até 1). 1 = sem ruído, 0 = ruído maximo
	for (int i = 0; i < src4.rows; i++) {//percorrer todas as linhas da imagem
		for (int a = 0; a < src4.cols; a = a++) {//percorrer todas as colunas da imagem
			pixel = src4.at<Vec3b>(i, a);//guarda a tonalidade de um pixel RGB(0 a 255,0 a 255,0 a 255) da imagem src4(original sem ruído)
			pixel1 = dst9.at<Vec3b>(i, a);//guarda a tonalidade de um pixel RGB(0 a 255,0 a 255,0 a 255) da imagem src(imagem ruído) (dst9 imagem com ruído após filtro guassiano blur)
			int blue = pixel[0];//guarda a tonalidade de azul na variavel blue (imagem original sem ruído)
			int green = pixel[1];//guarda a tonalidade de verde na variavel green(imagem original sem ruído)
			int red = pixel[2];//guarda a tonalidade de vermelho na variavel red(imagem original sem ruído)
			int blue1 = pixel1[0];//guarda a tonalidade de azul na variavel blue ("imagem ruído")
			int green1 = pixel1[1];//guarda a tonalidade de verde na variavel green("imagem ruído")
			int red1 = pixel1[2];//guarda a tonalidade de vermelho na variavel red("imagem ruído")
			if (alfa*blue + (1 - alfa)*blue1 > 255) {  //comparação da operação g(x) = (1 - alpha)h(x) + alpha f(x)
				src4.at<cv::Vec3b>(i, a)[0] = 255; // se a equação for maior que 255, colocamos ("set") em 255 a tonalidade azul do pixel tratado na iteração i (Modificação da imagem src4)
			}
			else
```

```
                                src4.at<cv::Vec3b>(i, a)[0] = alfa*blue + (1 - alfa)*blue1; //se a equação
não for maior, colocamos o resultado da equação na tonalidade de azul do pixel tratado na iteração i
(Modificação da imagem src4)
                        //mesmo procedimento abaixo, sendo que para as tonalidades green e red
                        if (alfa*green + (1 - alfa)*green1 > 255) {
                                src4.at<cv::Vec3b>(i, a)[1] = 255;
                        }
                        else
                                src4.at<cv::Vec3b>(i, a)[1] = alfa*green + (1 - alfa)*green1;
                        if (alfa*red + (1 - alfa)*red1 > 255) {
                                src4.at<cv::Vec3b>(i, a)[2] = 255;
                        }
                        else
                                src4.at<Vec3b>(i, a)[2] = alfa*red + (1 - alfa)*red1;
                }
        }
         imshow("Imagem somada", src4); //mostra o resultado da imagem somada (não é o resultado
do mixed das duas imagens)
//      imwrite("Imagens_Codigo_Completo/Imagem somada", src4);
                waitKey(0);
        for (int i = 1; i < src4.rows; i++) {//percorrer todas as linhas da imagem
                for (int a = 0; a < src4.cols; a = a++) {//percorrer todas as colunas da imagem
                        pixel = aux3.at<Vec3b>(i, a);//guarda a tonalidade de um pixel RGB(0 a 255,0 a
255,0 a 255) da imagem src4(original sem ruído).(utilizamos a variável aux3 pois src4 foi modificada)
                        pixel1 = src4.at<Vec3b>(i, a);//guarda a tonalidade de um pixel RGB(0 a 255,0 a
255,0 a 255) da imagem src4(imagem com ruído)
                        //mesmo procedimento utilizado anteriormente
                        int blue = pixel[0];
                        int green = pixel[1];
                        int red = pixel[2];
                        int blue1 = pixel1[0];
                        int green1 = pixel1[1];
                        int red1 = pixel1[2];
                        double Y = 0.299*red + 0.587*green + 0.114*blue;//guarda a luminância de um
pixel da imagem original sem ruído
                        double Y1 = 0.299*red1 + 0.587*green1 + 0.114*blue1;//guarda a luminância
de um pixel da imagem com ruído
                        if (Y <= Y1) {//comparação das luminâncias
                                //caso a luminância do pixel da imagem sem ruído for menor colocamos
suas tonalidades RGB na imagem final (imagem sem ruido + "imagem ruido")
                                src4.at<cv::Vec3b>(i, a)[0] = blue;//guardando tonalidade de azul do
pixel, da iteração i, de src4 em src4
                                src4.at<cv::Vec3b>(i, a)[1] = green;//guardando tonalidade de verde do
pixel , da iteração i ,de src4 em src4
                                src4.at<Vec3b>(i, a)[2] = red;//guardando a tonalidade de vermelhor do
pixel, da iteração i, de src4 em src4
                        }
                        else
                        {//caso a luminância do pixel da imagem sem ruído for maior, então colocamos
as tonalidades da imagem src4(imagem somada)
```

```
                                src4.at<cv::Vec3b>(i, a)[0] = blue1;
                                src4.at<cv::Vec3b>(i, a)[1] = green1;
                                src4.at<Vec3b>(i, a)[2] = red1;
                        }
                }
        }
        imshow("dark operation", src4);//Imagem final do mixed das duas imagens, (imagem sem ruído
+ "imagem ruído") com alfa = 0.5
        waitKey(0);
        imwrite("Imagens_Codigo_Completo/Imagem_Sintetizada.jpg", src4);
        // Gerar máscara para adicionar o envelhecimento:
        int NR = 153
        int NG = 233
        int NB = 252
        Mat img(1588, 1050, CV_8UC3, Scalar(NR, NG, NB)); //create an image ( 3 channels, 8 bit image
depth, 546 high, 728 wide, (0, 0, 50) assigned for Blue, Green and Red plane respectively. )
        namedWindow("Aging Mask", CV_WINDOW_AUTOSIZE); //create a window with the name
"MyWindow"
        imshow("Aging Mask", img); //display the image which is stored in the 'img' in the "MyWindow"
window
        waitKey(0);  //wait infinite time for a keypress
        destroyWindow("Aging Mask"); //destroy the window with the name, "MyWindow"
        // Mat dst9; //variável para armazenar a saída do filtro gaussiano
        char zBuffer8[35]; //variável para armazenar caracteres
        for (int i = 1; i < 5; i = i + 2)
        {
                _snprintf_s(zBuffer8, 35, "Kernel Size : %d x %d", i, i);//copia do texto "Kernel Size ..."
para a variável ZBuffer9
                // bilateralFilter(src, dst9, i, i * 4, i / 4);//filtro bilateral(imagem de entrada,imagem de
saída(mesmo tipo da imagem de entrada.Ex.: Mat,Iplimage...),diametro de cada vizinhança do
pixel,Desvio padrão no espaço de cor,Desvio padrao no espaço de coordenadas(em termo de pixel))
                GaussianBlur(img, img, Size(i, i), i * 2, i * 2);//filtro Gaussiano(imagem de
entrada,imagem de saída(mesmo tipo da imagem de entrada.Ex.: Mat,Iplimage...),Kernel Size, Gaussian
kernel standard deviation in X direction,Gaussian kernel standard deviation in Y direction)
                putText(img, zBuffer8, Point(img.cols / 128, img.rows / 32), FONT_HERSHEY_SIMPLEX,
0.4, Scalar(255, 159, 128), 1);//colocar o texto em uma imagem putText(imagem que irá colocar o texto,
texto que irá colocar tipo char ou string, coordenada onde colocará o texto, Fonte,Tamanho da Fonte,
Cor da fonte,tipo da linha(variável inteiro))
                imshow("Gaussian Blur Smoothed Image", img);//mostrar a saída após o filtro em cada
iteração
                int c = waitKey(2000); //espera dois segundos para proxima iteração
                kernel2 = i;
                if (c == 27)//if the "esc" key is pressed during the wait, return
                {
                        return;
                }
        }
        imshow("dark blur operation", img);//Imagem final do mixed das duas imagens, (imagem sem
ruído + "imagem ruído") com alfa = 0.5
        imwrite("Imagens_Codigo_Completo/Imagem_Sintetizada_Blur.jpg", img);
```

```
	waitKey(0);
	for (int i = 0; i < src4.rows; i++) {//percorrer todas as linhas da imagem
		for (int a = 0; a < src4.cols; a = a++) {//percorrer todas as colunas da imagem
			pixel = img.at<Vec3b>(i, a);//guarda a tonalidade de um pixel RGB(0 a 255,0 a
255,0 a 255) da imagem src4(original sem ruído).(envelhecimento)
			pixel1 = src4.at<Vec3b>(i, a);//guarda a tonalidade de um pixel RGB(0 a 255,0 a
255,0 a 255) da imagem src4(texo frente + verso)
			//mesmo procedimento utilizado anteriormente
			int blue = pixel[0];
			int green = pixel[1];
			int red = pixel[2];
			int blue1 = pixel1[0];
			int green1 = pixel1[1];
			int red1 = pixel1[2];
			if (blue + blue1 < 255) {
				src4.at<cv::Vec3b>(i, a)[0] = blue1;
			}
			else
				src4.at<cv::Vec3b>(i, a)[0] = blue + blue1;
			if (green + green1 < 255) {
				src4.at<cv::Vec3b>(i, a)[1] = green1;
			}
			else
				src4.at<cv::Vec3b>(i, a)[1] = green + green1;
			if (red + red1 < 255) {
				src4.at<Vec3b>(i, a)[2] = red1;
			}
			else
				src4.at<cv::Vec3b>(i, a)[2] = red + red1;
		}
	}
	imshow("dark operation = imagem sintetizada", src4);//Imagem final do mixed das duas
imagens, (imagem sem ruído + "imagem ruído") com alfa =
	imwrite("Imagens_Codigo_Completo/Imagem_Sintetizada.jpg", src4);
	waitKey(0);
	Mat dst3; //variável para armazenar a saída do filtro bilateral
	char zBuffer3[35]; //variável para armazenar caracteres
	for (int i = 1; i < 33; i = i + 2)
	{
		_snprintf_s(zBuffer3, 35, "Kernel Size : %d x %d", i, i);//copia do texto "Kernel Size ..."
para a variável ZBuffer3
		bilateralFilter(src4, dst3, i, i * 4, i / 4);//filtro bilateral(imagem de entrada,imagem de
saída(mesmo tipo da imagem de entrada.Ex.: Mat,Iplimage...),diametro de cada vizinhança do
pixel,Desvio padrão no espaço de cor,Desvio padrao no espaço de coordenadas(em termo de pixel))
		//GaussianBlur(src4, dst3, Size(i, i), 0, 0);//filtro Gaussiano(imagem de entrada,imagem
de saída(mesmo tipo da imagem de entrada.Ex.: Mat,Iplimage...),Kernel Size, Gaussian kernel standard
deviation in X direction,Gaussian kernel standard deviation in Y direction)
		putText(dst3, zBuffer3, Point(src4.cols / 128, src4.rows / 32), FONT_HERSHEY_SIMPLEX,
0.4, Scalar(255, 159, 128), 1);//colocar o texto em uma imagem putText(imagem que irá colocar o texto,
```

```
texto que irá colocar tipo char ou string, coordenada onde colocará o texto, Fonte,Tamanho da Fonte,
Cor da fonte,tipo da linha(variável inteiro))
                imshow("Smoothed Image", dst3);//mostrar a saída após o filtro em cada iteração
                int c = waitKey(2000); //espera dois segundos para proxima iteração
                kernel1 = i;
                if (c == 27)//if the "esc" key is pressed during the wait, return
                {
                        return;
                }
        }
        cout << " valor de alfa :" << alfa; //pular linha
        cout << " valor do kernel Blur: " << kernel; // exibe a variável kernel Blur
        cout << " valor do kernel (filter) \n " << kernel1; // exibe a variável kernel filter
        imwrite("Imagens_Codigo_Completo/Saída_do_Filtro.jpg", dst3); //salva a imagem dst3 no
diretorio do projeto com o nome dst3.jpg
        IplImage* img1 = cvLoadImage("Imagens_Codigo_Completo/Saída_do_Filtro.jpg");//carregar a
imagem dst3.jpg dentro do diretorio do projeto
        cvNamedWindow("Bilateral Filter Smoothed Image");//nome da janela da imagem (essa linha e
a posterior pode ser substituído por imshow)
        cvShowImage("Bilateral Filter Smoothed Image", img1);//mostra a saída do filtro (imagem
igual(dst3) a ultima iteração do laço(for) acima)
        //cvNamedWindow("Gaussian Filter Smoothed Image");//nome da janela da imagem (essa linha
e a posterior pode ser substituído por imshow)
        //cvShowImage("Gaussian Filter Smoothed Image", img1);//mostra a saída do filtro (imagem
igual(dst3) a ultima iteração do laço(for) acima)
        cvWaitKey(0);
        //Now, we'll create 3 grayscale images. These 3 images will hold the red, green and blue
channels of the image img1. Add these lines to the main function:
        IplImage* channelRed = cvCreateImage(cvGetSize(img1), 8, 1);
        IplImage* channelGreen = cvCreateImage(cvGetSize(img1), 8, 1);
        IplImage* channelBlue = cvCreateImage(cvGetSize(img1), 8, 1);
        cvSplit(img1, channelBlue, channelGreen, channelRed, NULL);//função para obtenção das
imagens no canal RGB
        //mostra as imagens nos canais RGB
        cvShowImage("Red channel", channelRed);
        cvShowImage("Green channel", channelGreen);
        cvShowImage("Blue channel", channelBlue);
        cvWaitKey(0);
        //Converter variável do tipo IplImage para Mat, para utilização de funções.
        //Converção dos 3 canais
        Mat imgBlue = cvarrToMat(channelBlue);
        Mat imgRed = cvarrToMat(channelRed);
        Mat imgGreen = cvarrToMat(channelGreen);
        imwrite("Imagens_Codigo_Completo/Canal_Blue.jpg", imgBlue);
        imwrite("Imagens_Codigo_Completo/Canal_Red.jpg", imgRed);
        imwrite("Imagens_Codigo_Completo/Canal_Green.jpg", imgGreen);
        int nPixels = imgBlue.total(); // função que calcula o numero total de pixels
        int nPixels1 = imgRed.total(); // função que calcula o numero total de pixels
        int nPixels2 = imgGreen.total(); // função que calcula o numero total de pixels
        int numBins = 256;//tons de cinzas (0 a 255)
```

```
	float range[] = { 0, 255 };//vetor utilizado para variação dos tons(0 a 255)
	const float *ranges[] = { range };//ponteiro para um vetor de float
	MatND histBlue, histRed, histGreen; //variáveis onde serão armazenadas os dados dos
histogramas
	//função para calcular os pontos do histogramas
	calcHist(&imgBlue, 1, 0, Mat(), histBlue, 1, &numBins, ranges, true, false);
	calcHist(&imgRed, 1, 0, Mat(), histRed, 1, &numBins, ranges, true, false);
	calcHist(&imgGreen, 1, 0, Mat(), histGreen, 1, &numBins, ranges, true, false);
	int auxBLUE = 0; //variável auxiliar, armazena a soma dos pixels
	int auxRED = 0;//variável auxiliar, armazena a soma dos pixels
	int auxGREEN = 0;//variável auxiliar, armazena a soma dos pixels
	int picoRED = 0;//variável para armazenar o maior número de pixel em um tom de cinza(0 a 255)
	int picoGREEN = 0; // variável para armazenar o maior número de pixel em um tom de cinza(0 a
255)
	int picoBLUE = 0; // variável para armazenar o maior número de pixel em um tom de cinza(0 a
255)
	//laço para soma dos pixels dos histogramas
	for (int h = 0; h < numBins; h++)
	{
		float binValBLUE = histBlue.at<float>(h); // variável binVal, armazena o valor do número
de pixels da iteração h
		float binValRED = histRed.at<float>(h);
		float binValGREEN = histGreen.at<float>(h);
		cout << " \n" << h; //pular linha
		cout << " " << binValBLUE; // exibe a variável binVal
		cout << " " << binValRED; // exibe a variável binVal
		cout << " " << binValGREEN; // exibe a variável binVal
		auxBLUE += binValBLUE; // variável auxiliar aux, armazena a soma dos pixels
		auxRED += binValRED; // variável auxiliar aux1, armazena a soma dos pixels
		auxGREEN += binValGREEN; // variável auxiliar aux1, armazena a soma dos pixels
		if (picoRED < binValRED) //comparação para otenção do pico
			picoRED = binValRED;
		if (picoGREEN < binValGREEN) //comparação para otenção do pico
			picoGREEN = binValGREEN;
		if (picoBLUE < binValBLUE) //comparação para otenção do pico
			picoBLUE = binValBLUE;
	}
	printf("\n");
	printf("\n");
	printf("Funcao do OPENCV\n");
	printf("Channel Red: %d\n", nPixels1); // exibe a quantidade total de pixels da imagem, através
da função do opencv img1.total();
	printf("Channel Green: %d\n", nPixels2); // exibe a quantidade total de pixels da imagem,
através da função do opencv img1.total();
	printf("Channel Blue: %d\n", nPixels); // exibe a quantidade total de pixels da imagem, através
da função do opencv img1.total();
	printf("\n");
	printf("\n");
	printf("Soma dos pixels\n");
	printf("Channel Red: %d\n", auxRED); // exibe a soma de pixels armazenada na variável aux.
```

```
        printf("Channel Green: %d\n", auxGREEN); // exibe a soma de pixels armazenada na variável
aux.
        printf("Channel Blue: %d\n", auxBLUE); // exibe a soma de pixels armazenada na variável aux.
        printf("PICO RED : %d\n", picoRED); // exibe a soma de pixels armazenada na variável aux.
        waitKey(0);
        // Imagem Original bmp
        Mat countpixelOriginalbmp = imread("IMF.bmp",
CV_LOAD_IMAGE_UNCHANGED);//carregando imagem sem ruido (bmp) e salvando na variável
countpixelOriginalbmp
        // cvShowImage("Original bmp", countpixelOriginalbmp);
        imwrite("Imagens_Codigo_Completo/Canal_Original_Binarizado.bmp", countpixelOriginalbmp);
        float countpixelblackOriginalbmp = 0;
        float countpixelwhiteOriginalbmp = 0;
        for (int i = 0; i < countpixelOriginalbmp.rows; i++) {//eixo x
                for (int a = 0; a < countpixelOriginalbmp.cols; a++) {//eixo y
                        if (countpixelOriginalbmp.at<uchar>(Point(a, i)) == 0)
                                countpixelblackOriginalbmp++;
                        else
                                countpixelwhiteOriginalbmp++;
                }
        }
        // Imagem Original jpg RGB
        cvThreshold(src42, src42, 100, 255, CV_THRESH_BINARY);
        cvShowImage("Original Threshold jpg RGB", src42);
        waitKey(0);
        Mat countpixelOriginal = cvarrToMat(src42);
        imwrite("Imagens_Codigo_Completo/Canal_Original_Binarizado.jpg", countpixelOriginal);
        float countpixelblackOriginal = 0;
        float countpixelwhiteOriginal = 0;
        for (int i = 0; i < countpixelOriginal.rows; i++) {//eixo x
                for (int a = 0; a < countpixelOriginal.cols; a++) {//eixo y
                        if (countpixelOriginal.at<uchar>(Point(a, i)) == 0)
                                countpixelblackOriginal++;

                        else
                                countpixelwhiteOriginal++;
                }
        }
        printf("\n");
        float  perOriginalWhitebmp;
        float  perOriginalBlackbmp;
        printf("Canal Original bmp\n");
        printf("Numero de pixel brancos(Original bmp): %.2f\n", countpixelwhiteOriginalbmp);
        printf("Numero de pixel pretos(Original bmp): %.2f\n", countpixelblackOriginalbmp);
        printf("SOMA DOS PIXELS CANAL Original %.2f\n", (countpixelwhiteOriginalbmp +
countpixelblackOriginalbmp));
        perOriginalWhitebmp = countpixelwhiteOriginalbmp * 100 / (countpixelwhiteOriginalbmp +
countpixelblackOriginalbmp);
        perOriginalBlackbmp = 100 - perOriginalWhitebmp;
        printf("Percentual de pixel brancos(Original bmp): %.2f\n", perOriginalWhitebmp);
```

```
        printf("Percentual de pixel pretos(Original bmp): %.2f\n", perOriginalBlackbmp);
        waitKey();
        printf("\n");
        printf("\n");
        printf("\n");
        float  perOriginalWhite;
        float  perOriginalBlack;
        printf("Canal Original jpg RGB\n");
        printf("Numero de pixel brancos(Original jpg RGB): %.2f\n", countpixelwhiteOriginal);
        printf("Numero de pixel pretos(Originaljpg RGB): %.2f\n", countpixelblackOriginal);
        printf("SOMA DOS PIXELS CANAL Original jpg RGB %.2f\n", (countpixelwhiteOriginal +
countpixelblackOriginal));
        perOriginalWhite = countpixelwhiteOriginal * 100 / (countpixelwhiteOriginal +
countpixelblackOriginal);
        perOriginalBlack = 100 - perOriginalWhite;
        printf("Percentual de pixel brancos(Original jpg RGB): %.2f\n", perOriginalWhite);
        printf("Percentual de pixel pretos(Original jpg RGB): %.2f\n", perOriginalBlack);
        waitKey();
        printf("\n");
        printf("\n");
        ofstream makefile;
        makefile.open("Dados do Código do Código Completo.txt");
        makefile << "Dados da Simulação";
        makefile << "\n";
        makefile << "Valor de Alpha : " << alfa;
        makefile << "\n";
        makefile << "Valor do kernel no verso (Blur): " << kernel;
        makefile << "\n";
        makefile << "Máscara Red do envelhecimento: " << NR;
        makefile << "\n";
        makefile << "Máscara Green do envelhecimento: " << NG;
        makefile << "\n";
        makefile << "Máscara Blue do envelhecimento: " << NB;
        makefile << "\n";
        makefile << "Valor do kernel no envelhecimento (Blur) : " << kernel2;
        makefile << "\n";
        makefile << "Valor do kernel (bilateral filter) : " << kernel1;
        makefile << "\n";
        makefile << "\n";
        makefile << "Canal Original bmp";
        makefile << "\n";
        makefile << "Numero de pixel brancos (channel Original bmp) : " <<
countpixelwhiteOriginalbmp;
        makefile << "\n";
        makefile << "Numero de pixel pretos(channel Original bmp): " << countpixelblackOriginalbmp;
        makefile << "\n";
        makefile << "SOMA DOS PIXELS CANAL Original bmp: " << (countpixelwhiteOriginalbmp +
countpixelblackOriginalbmp);
        makefile << "\n";
        makefile << "Percentual de pixel brancos(channel Original bmp): " << perOriginalWhitebmp;
```

```
	makefile << "\n";
	makefile << "Percentual de pixel pretos(channel Original bmp): " << perOriginalBlackbmp;
	makefile << "\n";
	makefile << "\n";
	makefile << "Canal Original jpg";
	makefile << "\n";
	makefile << "Numero de pixel brancos (channel Original jpg RGB) : " << countpixelwhiteOriginal;
	makefile << "\n";
	makefile << "Numero de pixel pretos(channel Original jpg RGB): " << countpixelblackOriginal;
	makefile << "\n";
	makefile << "SOMA DOS PIXELS CANAL Original jpg RGB: " << (countpixelwhiteOriginal +
countpixelblackOriginal);
	makefile << "\n";
	makefile << "Percentual de pixel brancos(channel Original jpg RGB): " << perOriginalWhite;
	makefile << "\n";
	makefile << "Percentual de pixel pretos(channel Original jpg RGB): " << perOriginalBlack;
	makefile << "\n";
	makefile << "\n";
	int modaRED = 0;//calcular a moda no canal Red
	int valordatonalidadeRED;//variável para armazenar o valor da tonalidade cinza no canal RED
	//laço para cálculo da moda
	for (float h = 0; h < numBins; h++)
	{
		float binVal1 = histRed.at<float>(h);// variável binVal, armazena o valor do número de
pixels da iteração h
		//	cout << " \n" << h; //pular linha
		// cout << " " << binVal1; // exibe a variável binVal
		modaRED += binVal1; // variável auxiliar armazena a soma dos pixels
		//	printf("\n");
		//	printf("SOMA\n");
		//	cout << " " << modaRED;
		if (modaRED > (auxRED / 2)) {//comparação entre o valor acumulado na moda e a
metade dos pixels da imagem
			valordatonalidadeRED = int(h*perOriginalWhite / 100);//armazena o valor do
tom de cinza caso a moda seja maior que a metade dos pixels
			break; //sair do laço
		}
	}
	valordatonalidadeRED = 130;

	printf("\n");
	printf("VALOR DE SOMA DOS PIXELS RED DIVIDO POR 2: %d\n", auxRED / 2);
	printf("valordatonalidade: %d\n", valordatonalidadeRED);
	cvWaitKey(0);
	//mesmo procedimento para o canal GREEN
	int modaGREEN = 0;
	int valordatonalidadeGREEN;
	for (int h = 0; h < numBins; h++)
	{
		float binVal1 = histGreen.at<float>(h);
```

```
//		cout << " \n" << h; //pular linha
//	cout << " " << binVal1; // exibe a variável binVal
modaGREEN += binVal1; // variável auxiliar armazena a soma dos pixels
//		printf("\n");
//		printf("SOMA\n");
//		cout << " " << modaGREEN;
if (modaGREEN > (auxGREEN / 2)) {
		valordatonalidadeGREEN = int(h*perOriginalWhite / 100);
		break;
}
}
valordatonalidadeGREEN = 130;
printf("\n");
printf("VALOR DE SOMA DOS PIXELS RED DIVIDO POR 2: %d\n", auxGREEN / 2);
printf("valordatonalidade: %d\n", valordatonalidadeGREEN);
cvWaitKey(0);
//mesmo procedimento para o canal BLUE
int modaBLUE = 0;
int valordatonalidadeBLUE;
for (int h = 0; h < numBins; h++)
{
		float binVal1 = histBlue.at<float>(h);
		//		cout << " \n" << h; //pular linha
		//		cout << " " << binVal1; // exibe a variável binVal
		modaBLUE += binVal1; // variável auxiliar armazena a soma dos pixels
		//		printf("\n");
		//		printf("SOMA\n");
		//		cout << " " << modaBLUE;
		if (modaBLUE > (auxBLUE / 2)) {
			valordatonalidadeBLUE = int(h*perOriginalWhite / 100);
			break;
		}
}
valordatonalidadeBLUE = 130;
printf("\n");
printf("VALOR DE SOMA DOS PIXELS RED DIVIDO POR 2: %d\n", auxBLUE / 2);
printf("valordatonalidade: %d\n", valordatonalidadeBLUE);
cvWaitKey(0);
int hist_w = 512; int hist_h = 128; //tamanho da janela dos histogramas (hist_w = largura e
hist_h = altura)
int bin_w = cvRound((double)hist_w / numBins);//fator de dimensionamento dos tons(0 a 255)
na janela de largura hist_w
Mat histImageRED(hist_h, hist_w, CV_8UC4, Scalar(255, 255, 255));//variável que armazenará a
imagem do histograma RED
normalize(histRed, histRed, 0, histImageRED.rows, NORM_MINMAX, -1, Mat());//Normalizes the
norm or value range of an array
//laço para desenhar as linhas do histograma
for (int i = 1; i < numBins; i++)
{
		line(histImageRED, Point(bin_w*(i - 1), hist_h - cvRound(histRed.at<float>(i - 1))),
```

```
                    Point(bin_w*(i), hist_h - cvRound(histRed.at<float>(i))),
                    Scalar(0, 0, 0), 2, 8, 0);
        }
        //imshow("Red Histogram", histImageRED); //histograma montado
        // waitKey(0);
        line(histImageRED, Point(valordatonalidadeRED * 2, 0), Point(valordatonalidadeRED * 2,
histImageRED.rows), Scalar(0, 0, 255), 2, 8);//linha threshold (coordenada multiplicada por 2 pela
largura ser de 512 = 2*256)
        line(histImageRED, Point(0, 0), Point(0, histImageRED.rows), Scalar(0, 0, 0), 2, 8);//linha para
formação do eixo y(quantidade de pixels)
        line(histImageRED, Point(0, histImageRED.rows), Point(histImageRED.cols, histImageRED.rows),
Scalar(0, 0, 0), 5, 8);//linha para formação do eixo x(tons de cinza_)
        stringstream ResultadoRED;//variável auxliar para converção do valor da tonalidade o qual é do
tipo int para o tipo string para utilizar a função putText
        ResultadoRED << valordatonalidadeRED;//resultadoRED recebe o valor da variável
valordatonalidadeRED
        putText(histImageRED, ResultadoRED.str(), Point(valordatonalidadeRED * 1.8,
histImageRED.rows / 5), FONT_HERSHEY_SIMPLEX, 0.4, Scalar(0, 0, 0), 1, 8);//coloca o valor da
tonalidade no histograma
        //laço para colocar os valores dos tons (eixo x) aos passos de 30.
        for (int i = 0; i < histImageRED.cols / 2; i = i + 25) {
                stringstream Resultado;
                Resultado << i;
                if (i == 0)
                        putText(histImageRED, Resultado.str(), Point(4, histImageRED.rows / 1.04),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if (i<236)
                        putText(histImageRED, Resultado.str(), Point(i * 2, histImageRED.rows / 1.04),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
        }
        putText(histImageRED, "255", Point(255 * 2 - 16, histImageRED.rows / 1.04),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
        int auxhistRED = 0;
        //laço para colocar o número de pixels (eixo y,divido em 5 partes)
        for (int i = 0; i < picoRED; i = i + picoRED / 5) {
                stringstream Resultado;
                Resultado << i;
                if (i == 0)
                        putText(histImageRED, "-", Point(0, histImageRED.rows - auxhistRED),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if (i>4 * picoRED / 5)
                        putText(histImageRED, Resultado.str(), Point(4, histImageRED.rows - auxhistRED
+ 4), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if ((i < picoRED) && (auxhistRED < histImageRED.rows))
                        putText(histImageRED, Resultado.str(), Point(4, histImageRED.rows -
auxhistRED), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                auxhistRED = auxhistRED + histImageRED.rows / 5;
        }
        stringstream picoREDaux;
        picoREDaux << picoRED;
```

```
        //putText(histImageRED, picoREDaux.str(), Point(4,0), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0,
0), 1, 8);
        imshow("Red Histogram Linha de Threshold", histImageRED);
        imwrite("Imagens_Codigo_Completo/Histograma_Canal_RED.jpg", histImageRED);
        //mesmo procedimento paro o canal GREEN
        Mat histImageGREEN(hist_h, hist_w, CV_8UC4, Scalar(255, 255, 255));
        normalize(histGreen, histGreen, 0, histImageGREEN.rows, NORM_MINMAX, -1, Mat());
        for (int i = 1; i < numBins; i++)
        {
                line(histImageGREEN, Point(bin_w*(i - 1), hist_h - cvRound(histGreen.at<float>(i - 1))),
                        Point(bin_w*(i), hist_h - cvRound(histGreen.at<float>(i))),
                        Scalar(0, 0, 0), 2, 8, 0);
        }
        //imshow("Green Histogram", histImageGREEN);
        // waitKey(0);
        line(histImageGREEN, Point(valordatonalidadeGREEN * 2, 0), Point(valordatonalidadeGREEN * 2,
histImageGREEN.rows), Scalar(0, 0, 255), 2, 8);
        line(histImageGREEN, Point(0, 0), Point(0, histImageGREEN.rows), Scalar(0, 0, 0), 2, 8);
        line(histImageGREEN, Point(0, histImageGREEN.rows), Point(histImageGREEN.cols,
histImageGREEN.rows), Scalar(0, 0, 0), 5, 8);
        stringstream ResultadoGREEN;
        ResultadoGREEN << valordatonalidadeGREEN;
        putText(histImageGREEN, ResultadoGREEN.str(), Point(valordatonalidadeGREEN * 1.8,
histImageGREEN.rows / 5), FONT_HERSHEY_SIMPLEX, 0.4, Scalar(0, 0, 0), 1, 8);
        for (int i = 0; i < histImageGREEN.cols / 2; i = i + 25) {
                stringstream Resultado;
                Resultado << i;
                if (i == 0)
                        putText(histImageGREEN, Resultado.str(), Point(4, histImageGREEN.rows /
1.04), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if (i<250)
                        putText(histImageGREEN, Resultado.str(), Point(i * 2, histImageGREEN.rows /
1.04), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
        }
        putText(histImageGREEN, "255", Point(255 * 2 - 16, histImageGREEN.rows / 1.04),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
        int auxhistGREEN = 0;
        for (int i = 0; i < picoGREEN; i = i + picoGREEN / 5) {
                stringstream Resultado;
                Resultado << i;
                if (i == 0)
                        putText(histImageGREEN, "-", Point(4, histImageGREEN.rows - auxhistGREEN),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if (i>4 * picoGREEN / 5)
                        putText(histImageGREEN, Resultado.str(), Point(4, histImageGREEN.rows -
auxhistGREEN + 4), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if ((i < picoGREEN) && (auxhistGREEN < histImageGREEN.rows))
                        putText(histImageGREEN, Resultado.str(), Point(4, histImageGREEN.rows -
auxhistGREEN + 2), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                auxhistGREEN = auxhistGREEN + histImageGREEN.rows / 5;
```

```
        }
        stringstream picoGREENaux;
        picoGREENaux << picoGREEN;
        //putText(histImageGREEN, picoGREENaux.str(), Point(4,0), FONT_HERSHEY_SIMPLEX, 0.3,
Scalar(0, 0, 0), 1, 8);
        imshow("Green Histogram Linha de Threshold", histImageGREEN);
        imwrite("Imagens_Codigo_Completo/Histograma_Canal_GREEN.jpg", histImageGREEN);
        //mesmo procedimento paro o canal BLUE
        Mat histImageBLUE(hist_h, hist_w, CV_8UC4, Scalar(255, 255, 255));
        normalize(histBlue, histBlue, 0, histImageBLUE.rows, NORM_MINMAX, -1, Mat());
        for (int i = 1; i < numBins; i++)
        {
                line(histImageBLUE, Point(bin_w*(i - 1), hist_h - cvRound(histBlue.at<float>(i - 1))),
                        Point(bin_w*(i), hist_h - cvRound(histBlue.at<float>(i))),
                        Scalar(0, 0, 0), 2, 8, 0);
        }
        //imshow("Blue Histogram", histImageBLUE);
        // waitKey(0);
        line(histImageBLUE, Point(valordatonalidadeBLUE * 2, 0), Point(valordatonalidadeBLUE * 2,
histImageBLUE.rows), Scalar(0, 0, 255), 2, 8);
        line(histImageBLUE, Point(0, 0), Point(0, histImageBLUE.rows), Scalar(0, 0, 0), 2, 8);
        line(histImageBLUE, Point(0, histImageBLUE.rows), Point(histImageBLUE.cols,
histImageBLUE.rows), Scalar(0, 0, 0), 5, 8);
        stringstream ResultadoBLUE;
        ResultadoBLUE << valordatonalidadeBLUE;
        putText(histImageBLUE, ResultadoBLUE.str(), Point(valordatonalidadeBLUE * 1.8,
histImageBLUE.rows / 5), FONT_HERSHEY_SIMPLEX, 0.4, Scalar(0, 0, 0), 1, 8);
        for (int i = 0; i < histImageBLUE.cols / 2; i = i + 25) {
                stringstream Resultado;
                Resultado << i;
                if (i == 0)
                        putText(histImageBLUE, Resultado.str(), Point(4, histImageBLUE.rows / 1.04),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if (i<250)
                        putText(histImageBLUE, Resultado.str(), Point(i * 2, histImageBLUE.rows / 1.04),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
        }
        putText(histImageBLUE, "255", Point(255 * 2 - 16, histImageBLUE.rows / 1.04),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
        int auxhistBLUE = 0;
        for (int i = 0; i < picoBLUE; i = i + picoBLUE / 5) {
                stringstream Resultado;
                Resultado << i;
                if (i == 0)
                        putText(histImageBLUE, "-", Point(0, histImageBLUE.rows - auxhistBLUE),
FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if (i>4 * picoBLUE / 5)
                        putText(histImageBLUE, Resultado.str(), Point(4, histImageBLUE.rows -
auxhistBLUE + 5), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                else if ((i < picoBLUE) && (auxhistBLUE < histImageBLUE.rows))
```

```
                putText(histImageBLUE, Resultado.str(), Point(4, histImageBLUE.rows -
auxhistBLUE + 2), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0, 0, 0), 1, 8);
                auxhistBLUE = auxhistBLUE + histImageBLUE.rows / 5;
        }
        stringstream picoBLUEaux;
        picoBLUEaux << picoBLUE;
        //putText(histImageBLUE, picoBLUEaux.str(), Point(4,0), FONT_HERSHEY_SIMPLEX, 0.3, Scalar(0,
0, 0), 1, 8);
        imshow("Blue Histogram Linha de Threshold", histImageBLUE);
        imwrite("Imagens_Codigo_Completo/Histograma_Canal_BLUE.jpg", histImageBLUE);
        Size smallSize(600, 240);//Tamanho da janela de cada segmento
        vector<Mat> smallImages;//Vetor que armazena cada segmento da imagem original
        //Laço para captura dos segmentos
        for (int y = 0; y < imgRed.rows; y += smallSize.height)//laço utilizado para percorrer todas as
linhas
        {
                if ((y + smallSize.height) >(imgRed.rows))//comparação utilizada para saber se chegou
no final da imagem em relação as linhas
                {//caso sim
                        for (int x = 0; x < imgRed.cols; x += smallSize.width)//laço utilizado para
percorrer todas as colunas de cada linha
                        {
                                if ((x + smallSize.width) >(imgRed.cols))//comparação utilizada para
saber se chegou no final da imagem em relação as colunas
                                {//caso sim
                                        Rect rect = Rect(x, y, (imgRed.cols - x), (imgRed.rows -
y));//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os outros
parametros sao largura e altura respectivamente)
                                        smallImages.push_back(Mat(imgRed, rect));//armazenando o
segmento dentro do vetor de imagens smallImages
                                        //código abaixo utilizado para vizualização do segmento
                                        //Mat smallImage = Mat(src4, rect);//armazenamento do
segmento em mat uma variavel somente para uso da linha de código abaixo
                                        //imshow("Pedaço de Image 2", smallImage);//mostrar o
segmento capturado
                                        //waitKey(0);//utilizado para visualizar os segmentos
                                }
                                else {
                                        //caso não tenha chegado no final das colunas da imagem
                                        Rect rect = Rect(x, y, smallSize.width, (imgRed.rows -
y));//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os outros
parametros sao largura e altura respectivamente)
                                        smallImages.push_back(Mat(imgRed, rect));//armazenando o
segmento dentro do vetor de imagens smallImages
                                        //código abaixo utilizado para vizualização do segmento
                                        // Mat smallImage = Mat(src4, rect);
                                        // imshow("Pedaço de Image", smallImage);
                                        // waitKey(0);
                                }
                        }
```

```
                }
                else
                {//caso não tenha chegado no final das linhas
                        for (int x = 0; x < imgRed.cols; x += smallSize.width)//laço utilizado para percorre
todas as colunas da linha
                        {
                                if ((x + smallSize.width) >(imgRed.cols)) {//comparação utilizada para
saber se chegou no final das colunas
                                        Rect rect = Rect(x, y, (imgRed.cols - x),
smallSize.height);//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os
outros parametros sao largura e altura respectivamente)
                                        smallImages.push_back(Mat(imgRed, rect));//armazenado o
segmento dentro do vetor de imagens smallImage
                                        //código abaixo utilizado para vizualização do segmento
                                        //Mat smallImage = Mat(src4, rect);
                                        //imshow("Pedaço de Image 2", smallImage1);
                                        //waitKey(0);
                                }
                                else {
                                        Rect rect = Rect(x, y, smallSize.width,
smallSize.height);//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os
outros parametros sao largura e altura respectivamente)
                                        smallImages.push_back(Mat(imgRed, rect));//armazenado o
segmento dentro do vetor de imagens smallImage
                                        //código abaixo utilizado para vizualização do segmento
                                        //Mat smallImage = Mat(src4, rect);
                                        //imshow("Pedaço de Image", smallImage);
                                        //waitKey(0);
                                }
                        }
                }
        }
        //Código para visualizar e armazenar todas as imagens armazenada no vetor smallImages, ou
seja, todos os segmentos
        string folderName = "Segmentos_Red_Codigo_Completo";
        string folderCreateCommand = "mkdir " + folderName;
        system(folderCreateCommand.c_str());
        for (int i = 0; i <smallImages.size(); i++) {
                stringstream ss;
                string name = "segmento_";
                string type = ".jpg";
                ss << name << (i + 1) << type;
                string filename = ss.str();
                ss.str("");
                ss << folderName << "/" << name << (i + 1) << type;
                string fullPath = ss.str();
                ss.str("");
                imwrite(fullPath, smallImages[i]);
                //      imshow("hsv", smallImages[i]); // descomentar para visualizar
                //      waitKey(0); //Note: wait for user input for every image
```

```
	}
	//	cout << " \n" << smallImages.size();//utilizado para saber a quantidade de
imagens(segmentos) armazenado no vetor smallImages
	Mat combinedRED(imgRed.rows, imgRed.cols, smallImages[0].type());//variável criada para
combinar os segmentos
	for (size_t i = 0; i < smallImages.size(); i++)
	{
		for (int y = 0; y < imgRed.rows; y += smallSize.height)
		{
			if ((y + smallSize.height) >(imgRed.rows))
			{
				for (int x = 0; x < imgRed.cols; x += smallSize.width)
				{
					if ((x + smallSize.width) >(imgRed.cols)) {
						Mat roi = combinedRED(Rect(x, y, (imgRed.cols - x),
imgRed.rows - y));
						smallImages[i].copyTo(roi);
						i++;
					}
					else {
						Mat roi = combinedRED(Rect(x, y, smallSize.width,
(imgRed.rows - y)));
						smallImages[i].copyTo(roi);
						i++;
					}
				}
			}
			else {
				for (int x = 0; x < imgRed.cols; x += smallSize.width)
				{
					if ((x + smallSize.width) >(imgRed.cols)) {
						Mat roi = combinedRED(Rect(x, y, (imgRed.cols - x),
smallSize.height));
						smallImages[i].copyTo(roi);
						i++;
					}
					else {
						Mat roi = combinedRED(Rect(x, y, smallSize.width,
smallSize.height));
						smallImages[i].copyTo(roi);
						i++;
					}
				}
			}
		}
	}
	//	imshow("combined image Red", combinedRED);//mostra as imagem remontada
	//	waitKey();
	vector<Mat> smallImagesGreen;//Vetor que armazena cada segmento da imagem original
	//Laço para captura dos segmentos
```

```
	for (int y = 0; y < imgGreen.rows; y += smallSize.height)//laço utilizado para percorrer todas as
linhas
	{
		if ((y + smallSize.height) >(imgGreen.rows))//comparação utilizada para saber se chegou
no final da imagem em relação as linhas
		{//caso sim
			for (int x = 0; x < imgGreen.cols; x += smallSize.width)//laço utilizado para
percorrer todas as colunas de cada linha
			{
				if ((x + smallSize.width) >(imgGreen.cols))//comparação utilizada para
saber se chegou no final da imagem em relação as colunas
				{//caso sim
					Rect rect = Rect(x, y, (imgGreen.cols - x), (imgGreen.rows -
y));//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os outros
parametros sao largura e altura respectivamente)
					smallImagesGreen.push_back(Mat(imgGreen,
rect));//armazenando o segmento dentro do vetor de imagens smallImages
					//código abaixo utilizado para vizualização do segmento
					//Mat smallImage = Mat(src4, rect);//armazenamento do
segmento em mat uma variavel somente para uso da linha de código abaixo
					//imshow("Pedaço de Image 2", smallImage);//mostrar o
segmento capturado
					//waitKey(0);//utilizado para visualizar os segmentos
				}
				else {
					//caso não tenha chegado no final das colunas da imagem
					Rect rect = Rect(x, y, smallSize.width, (imgGreen.rows -
y));//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os outros
parametros sao largura e altura respectivamente)
					smallImagesGreen.push_back(Mat(imgGreen,
rect));//armazenando o segmento dentro do vetor de imagens smallImages
					//código abaixo utilizado para vizualização do segmento
					//Mat smallImage = Mat(src4, rect);
					//imshow("Pedaço de Image", smallImage);
					//waitKey(0);
				}
			}
		}
		else
		{//caso não tenha chegado no final das linhas
			for (int x = 0; x < imgGreen.cols; x += smallSize.width)//laço utilizado para
percorre todas as colunas da linha
			{
				if ((x + smallSize.width) >(imgGreen.cols)) {
					//comparação utilizada para saber se chegou no final das
colunas
					Rect rect = Rect(x, y, (imgGreen.cols - x),
smallSize.height);//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os
outros parametros sao largura e altura respectivamente)
```

```
                                        smallImagesGreen.push_back(Mat(imgGreen,
rect));//armazenado o segmento dentro do vetor de imagens smallImage
                                        //código abaixo utilizado para vizualização do segmento
                                        //Mat smallImage = Mat(src4, rect);
                                        //imshow("Pedaço de Image 2", smallImage1);
                                        //waitKey(0);
                                }
                                else {
                                        //Caso não tenha chegado no final das colunas
                                        Rect rect = Rect(x, y, smallSize.width,
smallSize.height);//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os
outros parametros sao largura e altura respectivamente)
                                        smallImagesGreen.push_back(Mat(imgGreen,
rect));//armazenado o segmento dentro do vetor de imagens smallImage
                                        //código abaixo utilizado para vizualização do segmento
                                        //Mat smallImage = Mat(src4, rect);
                                        //imshow("Pedaço de Image", smallImage);
                                        //waitKey(0);
                                }
                        }
                }
        }
        //Código para visualizar e armazenar todas as imagens armazenada no vetor smallImages, ou
seja, todos os segmentos
        string folderName1 = "Segmentos_Green_Codigo_Completo";
        string folderCreateCommand1 = "mkdir " + folderName1;
        system(folderCreateCommand1.c_str());
        for (int i = 0; i <smallImagesGreen.size(); i++) {
                stringstream ss;
                string name = "segmento_";
                string type = ".jpg";
                ss << name << (i + 1) << type;
                string filename = ss.str();
                ss.str("");
                ss << folderName1 << "/" << name << (i + 1) << type;
                string fullPath = ss.str();
                ss.str("");
                imwrite(fullPath, smallImagesGreen[i]);
                //      imshow("hsv", smallImages[i]); // descomentar para visualizar
                //      waitKey(0); //Note: wait for user input for every image
        }
        //      cout << " \n" << smallImages.size();//utilizado para saber a quantidade de
imagens(segmentos) armazenado no vetor smallImages
        Mat combinedGREEN(imgGreen.rows, imgGreen.cols, smallImagesGreen[0].type());//variável
criada para combinar os segmentos
        //código abaixo para remontar a imagem original
        for (size_t i = 0; i < smallImagesGreen.size(); i++)
        {
                for (int y = 0; y < imgGreen.rows; y += smallSize.height)
                {
```

```
                        if ((y + smallSize.height) >(imgGreen.rows))
                        {
                                for (int x = 0; x < imgGreen.cols; x += smallSize.width)
                                {
                                        if ((x + smallSize.width) >(imgGreen.cols)) {
                                                Mat roi = combinedGREEN(Rect(x, y, (imgGreen.cols -
x), imgGreen.rows - y));
                                                smallImagesGreen[i].copyTo(roi);
                                                i++;
                                        }
                                        else {
                                                //cout << " \n" << x;//utilizado para debug
                                                //cout << " \n" << src4.cols;//utilizado para debug
                                                Mat roi = combinedGREEN(Rect(x, y, smallSize.width,
(imgRed.rows - y)));
                                                smallImagesGreen[i].copyTo(roi);
                                                i++;
                                        }
                                }
                        }
                        else {
                                for (int x = 0; x < imgGreen.cols; x += smallSize.width)
                                {
                                        if ((x + smallSize.width) >(imgGreen.cols)) {
                                                Mat roi = combinedGREEN(Rect(x, y, (imgGreen.cols -
x), smallSize.height));
                                                smallImagesGreen[i].copyTo(roi);
                                                i++;
                                        }
                                        else {
                                                //cout << " \n" << x;//utilizado para debug
                                                //cout << " \n" << src4.cols;//utilizado para debug
                                                Mat roi = combinedGREEN(Rect(x, y, smallSize.width,
smallSize.height));
                                                smallImagesGreen[i].copyTo(roi);
                                                i++;
                                        }
                                }
                        }
                }
        }
        //      imshow("combined image Green", combinedGREEN);//mostra as imagem remontada
        //      waitKey(0);
        vector<Mat> smallImagesBlue;//Vetor que armazena cada segmento da imagem original
        //Laço para captura dos segmentos
        for (int y = 0; y < imgBlue.rows; y += smallSize.height)//laço utilizado para percorrer todas as
linhas
        {
                if ((y + smallSize.height) >(imgBlue.rows))//comparação utilizada para saber se chegou
no final da imagem em relação as linhas
```

```
{//caso sim
	for (int x = 0; x < imgBlue.cols; x += smallSize.width)//laço utilizado para
percorrer todas as colunas de cada linha
	{
		if ((x + smallSize.width) >(imgBlue.cols))//comparação utilizada para
saber se chegou no final da imagem em relação as colunas
		{//caso sim
			Rect rect = Rect(x, y, (imgBlue.cols - x), (imgBlue.rows -
y));//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os outros
parametros sao largura e altura respectivamente)
			smallImagesBlue.push_back(Mat(imgBlue,
rect));//armazenando o segmento dentro do vetor de imagens smallImages
			//código abaixo utilizado para vizualização do segmento
			//Mat smallImage = Mat(src4, rect);//armazenamento do
segmento em mat uma variavel somente para uso da linha de código abaixo
			//imshow("Pedaço de Image 2", smallImage);//mostrar o
segmento capturado
			//waitKey(0);//utilizado para visualizar os segmentos
		}
		else {
			//caso não tenha chegado no final das colunas da imagem
			Rect rect = Rect(x, y, smallSize.width, (imgBlue.rows -
y));//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os outros
parametros sao largura e altura respectivamente)
			smallImagesBlue.push_back(Mat(imgBlue,
rect));//armazenando o segmento dentro do vetor de imagens smallImages
			//código abaixo utilizado para vizualização do segmento
			//Mat smallImage = Mat(src4, rect);
			//imshow("Pedaço de Image", smallImage);
			//waitKey(0);
		}
	}
}
else
{//caso não tenha chegado no final das linhas
	for (int x = 0; x < imgBlue.cols; x += smallSize.width)//laço utilizado para
percorre todas as colunas da linha
	{
		if ((x + smallSize.width) >(imgBlue.cols)) {
			//comparação utilizada para saber se chegou no final das
colunas
			Rect rect = Rect(x, y, (imgBlue.cols - x),
smallSize.height);//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os
outros parametros sao largura e altura respectivamente)
			smallImagesBlue.push_back(Mat(imgBlue, rect));//armazenado
o segmento dentro do vetor de imagens smallImage
			//código abaixo utilizado para vizualização do segmento
			//Mat smallImage = Mat(src4, rect);
			//imshow("Pedaço de Image 2", smallImage1);
			//waitKey(0);
```

```
                                }
                                else {
                                        //Caso não tenha chegado no final das colunas
                                        Rect rect = Rect(x, y, smallSize.width,
smallSize.height);//dimensao e localização  do segmento da imagem original (x e y=coordenadas e os
outros parametros sao largura e altura respectivamente)
                                        smallImagesBlue.push_back(Mat(imgBlue, rect));//armazenado
o segmento dentro do vetor de imagens smallImage
                                        //código abaixo utilizado para vizualização do segmento
                                        //Mat smallImage = Mat(src4, rect);
                                        //imshow("Pedaço de Image", smallImage);
                                        //waitKey(0);
                                }
                        }
                }
        }
        //Código para visualizar e armazenar todas as imagens armazenada no vetor smallImages, ou
seja, todos os segmentos
        string folderName2 = "Segmentos_Blue_Codigo_Completo";
        string folderCreateCommand2 = "mkdir " + folderName2;
        system(folderCreateCommand2.c_str());
        for (int i = 0; i <smallImagesBlue.size(); i++) {
                stringstream ss;
                string name = "segmento_";
                string type = ".jpg";
                ss << name << (i + 1) << type;
                string filename = ss.str();
                ss.str("");
                ss << folderName2 << "/" << name << (i + 1) << type;
                string fullPath = ss.str();
                ss.str("");
                imwrite(fullPath, smallImagesBlue[i]);
                //      imshow("hsv", smallImages[i]); // descomentar para visualizar
                //      waitKey(0); //Note: wait for user input for every image
        }
        //      cout << " \n" << smallImages.size();//utilizado para saber a quantidade de
imagens(segmentos) armazenado no vetor smallImages
        Mat combinedBLUE(imgBlue.rows, imgBlue.cols, smallImagesBlue[0].type());//variável criada
para combinar os segmentos
        //código abaixo para remontar a imagem original
        for (size_t i = 0; i < smallImagesBlue.size(); i++)
        {
                for (int y = 0; y < imgBlue.rows; y += smallSize.height)
                {
                        if ((y + smallSize.height) >(imgBlue.rows))
                        {
                                for (int x = 0; x < imgBlue.cols; x += smallSize.width)
                                {
                                        if ((x + smallSize.width) >(imgBlue.cols)) {
```

```
                                        Mat roi = combinedBLUE(Rect(x, y, (imgBlue.cols - x),
imgBlue.rows - y));
                                        smallImagesBlue[i].copyTo(roi);
                                        i++;
                                }
                                else {
                                        //cout << " \n" << x;//utilizado para debug
                                        //cout << " \n" << src4.cols;//utilizado para debug
                                        Mat roi = combinedBLUE(Rect(x, y, smallSize.width,
(imgRed.rows - y)));
                                        smallImagesBlue[i].copyTo(roi);
                                        i++;
                                }
                        }
                }
                else {
                        for (int x = 0; x < imgBlue.cols; x += smallSize.width)
                        {
                                if ((x + smallSize.width) >(imgBlue.cols)) {
                                        Mat roi = combinedBLUE(Rect(x, y, (imgBlue.cols - x),
smallSize.height));
                                        smallImagesBlue[i].copyTo(roi);
                                        i++;
                                }
                                else {
                                        //cout << " \n" << x;//utilizado para debug
                                        //cout << " \n" << src4.cols;//utilizado para debug
                                        Mat roi = combinedBLUE(Rect(x, y, smallSize.width,
smallSize.height));
                                        smallImagesBlue[i].copyTo(roi);
                                        i++;
                                }
                        }
                }
        }
        }
        //      imshow("combined image Blue", combinedBLUE);//mostra as imagem remontada
        //      waitKey(0);
        IplImage* ipl_imgRed = cvCloneImage(&(IplImage)combinedRED);
        IplImage* ipl_imgGreen = cvCloneImage(&(IplImage)combinedGREEN);
        IplImage* ipl_imgBlue = cvCloneImage(&(IplImage)combinedBLUE);
        valordatonalidadeRED = 130;
        int thresholdRed = valordatonalidadeRED;
        cvThreshold(ipl_imgRed, ipl_imgRed, valordatonalidadeRED, 255, CV_THRESH_BINARY);
        cvShowImage("Red Threshold", ipl_imgRed);
        waitKey(0);
        valordatonalidadeGREEN = 130;
        int thresholdGreen = valordatonalidadeGREEN;
        cvThreshold(ipl_imgGreen, ipl_imgGreen, valordatonalidadeGREEN, 255, CV_THRESH_BINARY);
        cvShowImage("Green Threshold", ipl_imgGreen);
```

```
	waitKey(0);
	valordatonalidadeBLUE = 130;
	int thresholdBlue = valordatonalidadeBLUE;
	cvThreshold(ipl_imgBlue, ipl_imgBlue, valordatonalidadeBLUE, 255, CV_THRESH_BINARY);
	cvShowImage("Blue Threshold", ipl_imgBlue);
	waitKey(0);
	Mat countpixelRed = cvarrToMat(ipl_imgRed);
	imwrite("Imagens_Codigo_Completo/Canal_RED_Binarizado.jpg", countpixelRed);
	float countpixelblackred = 0;
	float countpixelwhitered = 0;
	float perRedWhite, perRedBlack;
	for (int i = 0; i < countpixelRed.rows; i++) {
		for (int a = 0; a < countpixelRed.cols; a++) {
			if (countpixelRed.at<uchar>(Point(a, i)) == 0)
				countpixelblackred++;
			else
				countpixelwhitered++;
		}
	}
	// Cálculo do número de mudanças dos pixels Brancos e Pretos
	//Mat countpixelRed = cvarrToMat(ipl_imgRed);
	//Mat countpixelOriginal = cvarrToMat(src42);
	//float countpixelblackOriginal = 0;
	//float countpixelwhiteOriginal = 0;
	//float countpixelblackred = 0;
	//float countpixelwhitered = 0;
	float countpixelwhiteblackR = 0;
	float countpixelwhitewhiteR = 0;
	float countpixelblackwhiteR = 0;
	float countpixelblackblackR = 0;
	//float valcountpixelOriginal = countpixelOriginal.at<float>(Point(a, i)); // variável
valcountpixelOriginal, armazena o valor do pixels da iteração a,i
	//float valcountpixelRed = countpixelRed.at<float>(Point(a, i)); // variável valcountpixelRed,
armazena o valor do pixels da iteração a,i
	//cout << " " << valcountpixelOriginal; // exibe a variável countpixelOriginal.at<uchar>(Point(a, i)
	//cout << " " << valcountpixelRed; // exibe a variável countpixelOriginal.at<uchar>(Point(a, i)
	//waitKey(0);
	for (int i = 0; i < countpixelOriginal.rows; i++) {//eixo x
		for (int a = 0; a < countpixelOriginal.cols; a++) {//eixo y
			if (countpixelOriginal.at<uchar>(Point(a, i)) > countpixelRed.at<uchar>(Point(a,
i)))
				countpixelwhiteblackR++;
			else if (countpixelOriginal.at<uchar>(Point(a, i)) <
countpixelRed.at<uchar>(Point(a, i)))
				countpixelblackwhiteR++;
			else
			if (countpixelRed.at<uchar>(Point(a, i)) == 0)
				countpixelblackblackR++;
			else
				countpixelwhitewhiteR++;
```

```
                }
            }
            printf("\n");
            printf("\n");
            printf("Canal RED\n");
            printf("Threshold (channel Red): %.2f\n", thresholdRed);
            printf("Numero de pixel brancos(channel Red): %.2f\n", countpixelwhitered);
            printf("Numero de pixel pretos(channel Red): %.2f\n", countpixelblackred);
            printf("SOMA DOS PIXELS CANAL RED %.2f\n", (countpixelwhitered + countpixelblackred));
            perRedWhite = countpixelwhitered * 100 / (countpixelwhitered + countpixelblackred);
            perRedBlack = 100 - perRedWhite;
            printf("Percentual de pixel brancos(channel Red): %.2f\n", perRedWhite);
            printf("Percentual de pixel pretos(channel Red): %.2f\n", perRedBlack);
            printf("\n");
            printf("\n");
            float  perWhiteWhiteR;
            float  perWhiteBlackR;
            float  perBlackBlackR;
            float  perBlackWhiteR;
            perWhiteWhiteR = countpixelwhitewhiteR * 100 / (countpixelwhitewhiteR +
countpixelwhiteblackR);
            perWhiteBlackR = countpixelwhiteblackR * 100 / (countpixelwhitewhiteR +
countpixelwhiteblackR);
            perBlackBlackR = countpixelblackblackR * 100 / (countpixelblackwhiteR +
countpixelblackblackR);
            perBlackWhiteR = countpixelblackwhiteR * 100 / (countpixelblackwhiteR +
countpixelblackblackR);
            makefile << "Canal RED";
            makefile << "\n";
            makefile << "Threshold (channel Red):" << thresholdRed;
            makefile << "\n";
            makefile << "Numero de pixel brancos (channel RED) : " << countpixelwhitered;
            makefile << "\n";
            makefile << "Numero de pixel pretos(channel RED): " << countpixelblackred;
            makefile << "\n";
            makefile << "SOMA DOS PIXELS CANAL RED: " << (countpixelwhitered + countpixelblackred);
            makefile << "\n";
            makefile << "Percentual de pixel brancos(channel Red): " << perRedWhite;
            makefile << "\n";
            makefile << "Percentual de pixel pretos(channel Red): " << perRedBlack;
            makefile << "\n";
            makefile << "\n";
            makefile << "Numero de pixels que eram brancos e continuaram brancos (channel RED) : " <<
countpixelwhitewhiteR;
            makefile << "\n";
            makefile << "Percentual de pixels que eram brancos e continuaram brancos (channel RED) : " <<
perWhiteWhiteR;
            makefile << "\n";
            makefile << "Numero de pixels que eram brancos e mudaram para pretos (channel RED) : " <<
countpixelwhiteblackR;
```

```
	makefile << "\n";
	makefile << "Percentual de pixels que eram brancos e mudaram para pretos (channel RED) : " <<
perWhiteBlackR;
	makefile << "\n";
	makefile << "Numero de pixels que eram pretos e continuaram pretos (channel RED) : " <<
countpixelblackblackR;
	makefile << "\n";
	makefile << "Percentual de pixels que eram pretos e continuaram pretos (channel RED) : " <<
perBlackBlackR;
	makefile << "\n";
	makefile << "Numero de pixels que eram pretos e mudaram para brancos (channel RED) : " <<
countpixelblackwhiteR;
	makefile << "\n";
	makefile << "Percentual de pixels que eram pretos e mudaram para brancos (channel RED) : " <<
perBlackWhiteR;
	makefile << "\n";
	makefile << "\n";
	Mat countpixelGreen = cvarrToMat(ipl_imgGreen);
	imwrite("Imagens_Codigo_Completo/Canal_GREEN_Binarizado.jpg", countpixelGreen);
	float countpixelblackgreen = 0;
	float countpixelwhitegreen = 0;
	float perGreenWhite, perGreenBlack;
	for (int i = 0; i < countpixelGreen.rows; i++) {
		for (int a = 0; a < countpixelGreen.cols; a++) {
			if (countpixelGreen.at<uchar>(Point(a, i)) == 0)
				countpixelblackgreen++;
			else
				countpixelwhitegreen++;
		}
	}
	float countpixelwhiteblackG = 0;
	float countpixelwhitewhiteG = 0;
	float countpixelblackwhiteG = 0;
	float countpixelblackblackG = 0;
	for (int i = 0; i < countpixelOriginal.rows; i++) {//eixo x
		for (int a = 0; a < countpixelOriginal.cols; a++) {//eixo y
			if (countpixelOriginal.at<uchar>(Point(a, i)) >
countpixelGreen.at<uchar>(Point(a, i)))
				countpixelwhiteblackG++;
			else if (countpixelOriginal.at<uchar>(Point(a, i)) <
countpixelGreen.at<uchar>(Point(a, i)))
				countpixelblackwhiteG++;
			else
			if (countpixelRed.at<uchar>(Point(a, i)) == 0)
				countpixelblackblackG++;
			else
				countpixelwhitewhiteG++;
		}
	}
	printf("Canal GREEN\n");
```

```
        printf("Threshold (channel Green): %.2f\n", thresholdGreen);
        printf("Numero de pixel brancos(channel Green): %.2f\n", countpixelwhitegreen);
        printf("Numero de pixel pretos(channel Green): %.2f\n", countpixelblackgreen);
        printf("SOMA DOS PIXELS CANAL Green %.2f\n", (countpixelwhitegreen +
countpixelblackgreen));
        perGreenWhite = countpixelwhitegreen * 100 / (countpixelwhitegreen + countpixelblackgreen);
        perGreenBlack = 100 - perGreenWhite;
        printf("Percentual de pixel brancos(channel Green): %.2f\n", perGreenWhite);
        printf("Percentual de pixel pretos(channel Green): %.2f\n", perGreenBlack);
        waitKey(0);
        printf("\n");
        printf("\n");
        float  perWhiteWhiteG;
        float  perWhiteBlackG;
        float  perBlackBlackG;
        float  perBlackWhiteG;
        perWhiteWhiteG = countpixelwhitewhiteG * 100 / (countpixelwhitewhiteG +
countpixelwhiteblackG);
        perWhiteBlackG = countpixelwhiteblackG * 100 / (countpixelwhitewhiteG +
countpixelwhiteblackG);
        perBlackBlackG = countpixelblackblackG * 100 / (countpixelblackwhiteG +
countpixelblackblackG);
        perBlackWhiteG = countpixelblackwhiteG * 100 / (countpixelblackwhiteG +
countpixelblackblackG);
        makefile << "Canal GREEN";
        makefile << "\n";
        makefile << "Threshold (channel Green):" << thresholdGreen;
        makefile << "\n";
        makefile << "Numero de pixel brancos (channel GREEN) : " << countpixelwhitegreen;
        makefile << "\n";
        makefile << "Numero de pixel pretos(channel GREEN): " << countpixelblackgreen;
        makefile << "\n";
        makefile << "SOMA DOS PIXELS CANAL GREEN: " << (countpixelwhitegreen +
countpixelblackgreen);
        makefile << "\n";
        makefile << "Percentual de pixel brancos(channel GREEN): " << perGreenWhite;
        makefile << "\n";
        makefile << "Percentual de pixel pretos(channel GREEN): " << perGreenBlack;
        makefile << "\n";
        makefile << "\n";
        makefile << "Numero de pixel que eram brancos e continuaram brancos (channel GREEN) : " <<
countpixelwhitewhiteG;
        makefile << "\n";
        makefile << "Percentual de pixels que eram brancos e continuaram brancos (channel GREEN) : "
<< perWhiteWhiteG;
        makefile << "\n";
        makefile << "Numero de pixel que eram brancos e mudaram para pretos (channel GREEN) : " <<
countpixelwhiteblackG;
        makefile << "\n";
```

```
		makefile << "Percentual de pixels que eram brancos e mudaram para pretos (channel GREEN) : "
<< perWhiteBlackG;
		makefile << "\n";
		makefile << "Numero de pixel que eram pretos e continuaram pretos (channel GREEN) : " <<
countpixelblackblackG;
		makefile << "\n";
		makefile << "Percentual de pixels que eram pretos e continuaram pretos (channel GREEN) : " <<
perBlackBlackG;
		makefile << "\n";
		makefile << "Numero de pixel que eram pretos e mudaram para brancos (channel GREEN) : " <<
countpixelblackwhiteG;
		makefile << "\n";
		makefile << "Percentual de pixels que eram pretos e mudaram para brancos (channel GREEN) : "
<< perBlackWhiteG;
		makefile << "\n";
		makefile << "\n";
		Mat countpixelBlue = cvarrToMat(ipl_imgBlue);
		imwrite("Imagens_Codigo_Completo/Canal_BLUE_Binarizado.jpg", countpixelBlue);
		float countpixelblackblue = 0;
		float countpixelwhiteblue = 0;
		float perBlueWhite, perBlueBlack;
		for (int i = 0; i < countpixelBlue.rows; i++) {
			for (int a = 0; a < countpixelBlue.cols; a++) {
				if (countpixelBlue.at<uchar>(Point(a, i)) == 0)
					countpixelblackblue++;
				else
					countpixelwhiteblue++;
			}
		}
		float countpixelwhiteblackB = 0;
		float countpixelwhitewhiteB = 0;
		float countpixelblackwhiteB = 0;
		float countpixelblackblackB = 0;
		for (int i = 0; i < countpixelOriginal.rows; i++) {//eixo x
			for (int a = 0; a < countpixelOriginal.cols; a++) {//eixo y
				if (countpixelOriginal.at<uchar>(Point(a, i)) > countpixelBlue.at<uchar>(Point(a,
i)))
					countpixelwhiteblackB++;
				else if (countpixelOriginal.at<uchar>(Point(a, i)) <
countpixelBlue.at<uchar>(Point(a, i)))
					countpixelblackwhiteB++;
				else
				if (countpixelRed.at<uchar>(Point(a, i)) == 0)
					countpixelblackblackB++;
				else
					countpixelwhitewhiteB++;
			}
		}
		printf("Canal Blue\n");
		printf("Threshold (channel Blue): %.2f\n", thresholdBlue);
```

```
        printf("Numero de pixel brancos(channel Blue): %.2f\n", countpixelwhiteblue);
        printf("Numero de pixel pretos(channel Blue): %.2f\n", countpixelblackblue);
        printf("SOMA DOS PIXELS CANAL BLUE %.2f\n", (countpixelwhiteblue + countpixelblackblue));
        perBlueWhite = countpixelwhiteblue * 100 / (countpixelwhiteblue + countpixelblackblue);
        perBlueBlack = 100 - perBlueWhite;
        printf("Percentual de pixel brancos(channel Blue): %.2f\n", perBlueWhite);
        printf("Percentual de pixel pretos(channel Blue): %.2f\n", perBlueBlack);
        printf("\n");
        printf("\n");
        float  perWhiteWhiteB;
        float  perWhiteBlackB;
        float  perBlackBlackB;
        float  perBlackWhiteB;
        perWhiteWhiteB = countpixelwhitewhiteB * 100 / (countpixelwhitewhiteB +
countpixelwhiteblackB);
        perWhiteBlackB = countpixelwhiteblackB * 100 / (countpixelwhitewhiteB +
countpixelwhiteblackB);
        perBlackBlackB = countpixelblackblackB * 100 / (countpixelblackwhiteB +
countpixelblackblackB);
        perBlackWhiteB = countpixelblackwhiteB * 100 / (countpixelblackwhiteB +
countpixelblackblackB);
        makefile << "Canal BLUE";
        makefile << "\n";
        makefile << "Threshold (channel Blue):" << thresholdBlue;
        makefile << "\n";
        makefile << "Numero de pixel brancos (channel BLUE) : " << countpixelwhiteblue;
        makefile << "\n";
        makefile << "Numero de pixel pretos(channel BLUE): " << countpixelblackblue;
        makefile << "\n";
        makefile << "SOMA DOS PIXELS CANAL BLUE: " << (countpixelwhiteblue + countpixelblackblue);
        makefile << "\n";
        makefile << "Percentual de pixel brancos(channel BLUE): " << perBlueWhite;
        makefile << "\n";
        makefile << "Percentual de pixel pretos(channel BLUE): " << perBlueBlack;
        makefile << "\n";
        makefile << "\n";
        makefile << "Numero de pixel que eram brancos e continuaram brancos (channel BLUE) : " <<
countpixelwhitewhiteB;
        makefile << "\n";
        makefile << "Percentual de pixels que eram brancos e continuaram brancos (channel BLUE) : "
<< perWhiteWhiteB;
        makefile << "\n";
        makefile << "Numero de pixel que eram brancos e mudaram para pretos (channel BLUE) : " <<
countpixelwhiteblackB;
        makefile << "\n";
        makefile << "Percentual de pixels que eram brancos e mudaram para pretos (channel BLUE) : "
<< perWhiteBlackB;
        makefile << "\n";
        makefile << "Numero de pixel que eram pretos e continuaram pretos (channel BLUE) : " <<
countpixelblackblackB;
```

```
        makefile << "\n";
        makefile << "Percentual de pixels que eram pretos e continuaram pretos (channel BLUE) : " <<
perBlackBlackB;
        makefile << "\n";
        makefile << "Numero de pixel que eram pretos e mudaram para brancos (channel BLUE) : " <<
countpixelblackwhiteB;
        makefile << "\n";
        makefile << "Percentual de pixels que eram pretos e mudaram para brancos (channel BLUE) : "
<< perBlackWhiteB;
        makefile << "\n";
        makefile << "\n";
        makefile.close();
        waitKey(0);
        return;
}
```

10 APPENDIX D

10.1 Others Figures

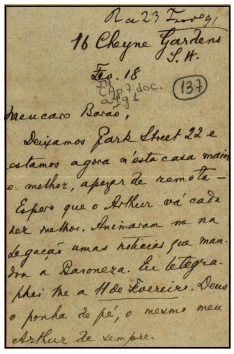


Figure 10.1: Historical Document 01

Table 10.1: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	180.181	11.802	182	129	255
Green	164.777	13.220	165	115	255
Blue	122.879	18.099	122	76	255
GrayScale Intensity	164.668	13.205	165	115	255

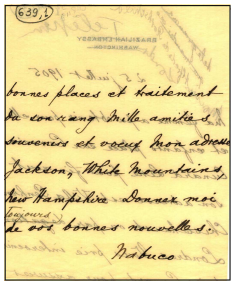


Figure 10.2: Historical Document 02

Table 10.2: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	253.425	1.933	255	210	255
Green	237.834	3.941	238	191	251
Blue	157.887	3.245	158	112	172
GrayScale Intensity	233.360	2.926	233	188	243

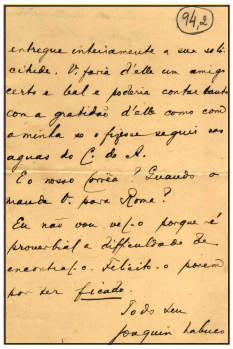


Figure 10.3: Historical Document 03

Table 10.3: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	241.029	4.426	241	182	255
Green	200.181	4.474	201	144	215
Blue	136.541	4.590	137	81	153
GrayScale Intensity	205.157	4.413	206	148	220

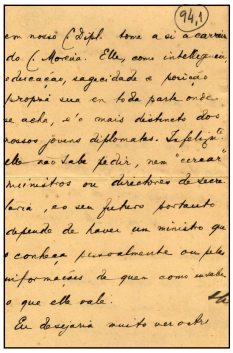


Figure 10.4: Historical Document 04

Table 10.4: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	245.117	5.201	247	171	255
Green	204.011	4.266	206	130	221
Blue	141.076	4.032	142	68	153
GrayScale Intensity	209.127	4.376	211	135	223

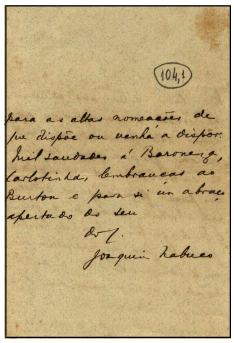


Figure 10.5: Historical Document 05

Table 10.5: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	200.766	7.779	203	157	223
Green	181.099	7.044	183	140	201
Blue	133.725	6.762	136	94	157
GrayScale Intensity	181.565	7.048	184	140	202

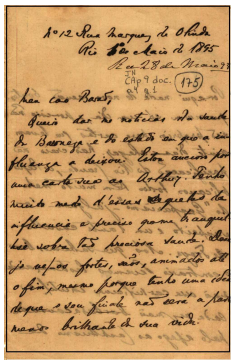


Figure 10.6: Historical Document 06

Table 10.6: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	224.024	4.131	225	164	238
Green	173.971	4.338	175	112	188
Blue	112.848	4.208	113	52	127
GrayScale Intensity	181.947	4.207	182	121	196

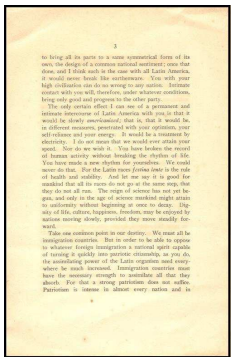


Figure 10.7: Historical Document 07

Table 10.7: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	255.001	1.354	255	213	255
Green	246.626	5.259	251	200	255
Blue	207.651	6.483	213	158	221
GrayScale Intensity	244.393	3.982	246	199	251

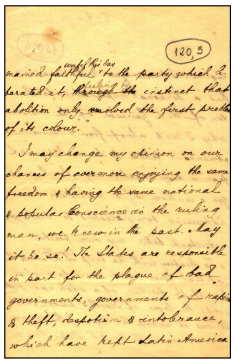


Figure 10.8: Historical Document 08

Table 10.8: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	252.760	2.278	255	238	255
Green	231.753	5.185	233	210	253
Blue	152.443	4.493	153	136	169
GrayScale Intensity	228.989	3.880	229	210	244

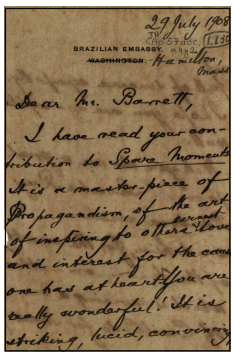


Figure 10.9: Historical Document 09

Table 10.9: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	154.017	6.988	155	115	179
Green	128.111	7.267	129	83	151
Blue	89.039	7.181	91	44	112
GrayScale Intensity	131.464	7.056	132	88	155

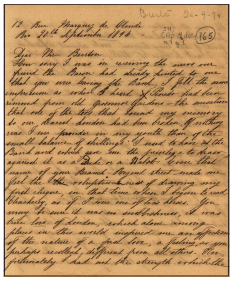
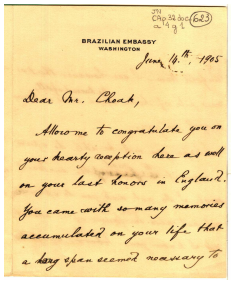


Figure 10.10: Historical Document 10

Table 10.10: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	217.885	6.282	221	174	232
Green	169.440	5.950	172	125	186
Blue	112.629	5.335	114	69	130
GrayScale Intensity	177.381	5.909	180	133	193


BRAZILIAN EMBASSY
WASHINGTON

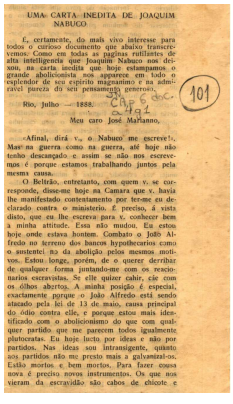
Dear Mr. Clark,

Allow me to congratulate you on your hearty reception here as well on your last honors in England. You came with so many memories accumulated on your life that a day when record necessary to

Figure 10.11: Historical Document 11

Table 10.11: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	252.559	2.863	255	232	255
Green	222.729	7.314	227	188	244
Blue	155.652	6.648	158	125	175
GrayScale Intensity	244.017	5.607	227	194	239


UMA CARTA INEDITA DE JOAQUIM NABUCO

E, certamente, do mais vivo interesse para todos o curioso documento que abaixo transcrevemos. Como em todas as paginas rutilantes de alta intelligencia que Joaquim Nabuco nos deixou, na carta inedita que hoje estampamos o grande abolicionista nos apparece em todo o resplendor de seu espirito magnanimo e na admiravel pureza de seu pensamento generoso.

Rio, Julho — 1888.

Meu caro José Mariano,

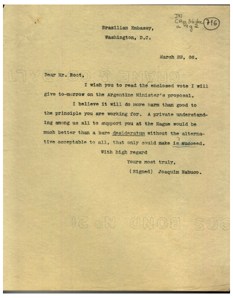
Afinal, dirá v., o Nabuco me escreve!... Mas na guerra como na guerra, até hoje não tenho descançado e assim se não nos escrevemos é porque estamos trabalhando juntos pela mesma causa.

O Beltrão, entretanto, com quem v. se corresponde, disse-me hoje na Camara que v. havia lhe manifestado contentamento por ter-me eu declarado contra o ministerio. É preciso, á vista disto, que eu lhe escreva para v. conhecer bem a minha attitude. Essa não mudou. Eu estou hoje onde estava hontem. Combato o João Alfredo no terreno dos bancos hypothecarios como o sustentei no da abolição pelos mesmos motivos. Estou longe, porém, de o querer derribar de qualquer forma juntando-me com os reaccionarios escravistas. Se elle quizer cahir, cáe com os olhos abertos. A minha posição é especial, exactamente porque o João Alfredo está sendo aniquilado pela lei de 13 de maio, causa principal do ódio contra elle, e porque estou mais identificado com o abolicionismo do que com qualquer partido que me parecem todos igualmente plutocratas. Eu hoje lucto por ideas e não por partidos. Nas ideas sou intransigente, quanto aos partidos não me presto mais a galvanisal-os. Estão mortos e bem mortos. Para fazer cousa nova é preciso novos instrumentos. Os que nos vieram da escravidão são cabos de chicote e

Figure 10.12: Historical Document 12

Table 10.12: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	247.773	5.229	255	221	255
Green	196.935	7.088	199	166	225
Blue	132.022	7.826	135	94	161
GrayScale Intensity	204.756	6.465	207	175	227



JN CAp 36 doc a 492 (716)

Brazilian Embassy,
Washington, D.C.

March 22, 06.

Dear Mr. Root,

I wish you to read the enclosed vote I will give to-morrow on the Argentine Minister's proposal.

I believe it will do more harm than good to the principle you are working for. A private understanding among us all to support you at the Hague would be much better than a bare desideratum without the alternative acceptable to all, that only could make is succeed.

With high regard

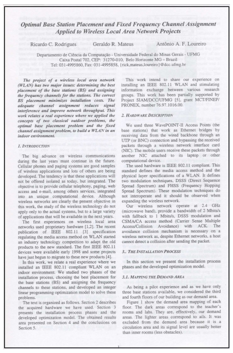
Yours most truly,

(Signed) Joaquim Nabuco.

Figure 10.13: Historical Document 13

Table 10.13: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	211.585	6.833	212	131	255
Green	192.345	6.580	193	113	245
Blue	140.739	6.390	141	63	186
GrayScale Intensity	192.162	6.552	193	113	241



Optimal Base Station Placement and Fixed Frequency Channel Assignment Applied to Wireless Local Area Network Projects

Ricardo C. Rodrigues Geraldo R. Mateus Antônio A. F. Loureiro

Departamento de Ciência da Computação - Universidade Federal do Minas Gerais - UFMG
Caixa Postal 702, CEP: 31270-010, Belo Horizonte MG – Brazil
Tel: 031-4995860, Fax: 031-4995858, {rick,mateus,loureiro}@dcc.ufmg.br

The project of a wireless local area network (WLAN) has two major issues: determining the best placement of the base stations (BS) and assigning the frequency channels for the stations. The correct BS placement minimizes installation costs. The adequate channel assignment reduces signal interference and improve network throughput. This work relates a real experience where we applied the concepts of two classical outdoor problems, the optimal base placement problem and the fixed channel assignment problem, to build a WLAN in an indoor environment.

1. Introduction

The big advance on wireless communications during the last years must continue in the future. Cellular phones and paging systems are good samples of wireless applications and lots of others are being developed. The tendency is that these applications will not be offered isolated as today, but integrated. The objective is to provide cellular telephony, paging, web access and e-mail, among others services, integrated into an unique computational device. Although wireless networks are clearly the present objective in this work, the study of the wireless technology do not apply only to the actual systems, but to a large variety of applications that will be available in the next years.

The first experiences on wireless local area networks used proprietary hardware [1,2]. The recent publication of IEEE 802.11 [3] specification regulating the media access method on WLAN started an industry technology competition to adapt the old products to the new standard. The first IEEE 802.11 devices were available early 1998 and some projects have just begun to migrate to these new products [4].

In this work, we relate a real experience where we installed an IEEE 802.11 compliant WLAN on an indoor environment. We studied two phases of the installation process, choosing the best placement for the base stations (BS) and assigning the frequency channels to these stations, and developed an integer linear programming optimization model to solve these problems.

The text is organized as follows. Section 2 describes the acquired hardware we have used. Section 3 presents the installation process phases and the developed optimization model. The obtained results area presented on Section 4 and the conclusions on Section 5.

This work intend to share our experience on installing an IEEE 802.11 WLAN and stimulating information exchange between various research groups. This work has been partially supported by Project SIAM/DCC/UFMG [5], grant MCT/FINEP/PRONEX, number 76.97.1016.00.

2. Hardware Description

We used three WavePOINT-II Access Points (the base stations) that work as Ethernet bridges by receiving data from the wired backbone through an UTP (or BNC) connection and bypassing the received packets through a wireless network interface card (NIC). The mobile users receive these packets through another NIC attached to its laptop or other computational device.

The used hardware is IEEE 802.11 compliant. This standard defines the media access method and the physical layer specifications of a WLAN. It defines two modulation techniques, DSSS (Direct Sequence Spread Spectrum) and FHSS (Frequency Hopping Spread Spectrum). These modulation techniques do not interoperate and it should be observed when expanding the wireless network.

Our wireless network operate at 2.4 GHz (microwave band), provide a bandwidth of 2 Mbits/s with fallback to 1 Mbits/s, DSSS modulation and CSMA/CA access method (Carrier Sense Multiple Access/Collision Avoidance) with ACK. The avoidance collision mechanism is necessary on a wireless network as, despite Ethernet networks, a host cannot detect a collision after sending the packet.

3.. The Installation Process

In this section we present the installation process phases and the developed optimization model.

3.1. Mapping the Demand Area

As being a pilot experience and as we have only three base stations available, we considered the third and fourth floors of our building as our demand area.

Figure 1 show the demand area mapping of each floor. The dark areas correspond to the teacher's rooms and labs. They are, effectively, our demand areas. The lighter areas correspond to ails. It was excluded from the demand area because it is a circulation area and its signal level are usually better than inner rooms (less obstacles).

Figure 10.14: Historical Document 14

Table 10.14: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	242.486	5.150	246	222	255
Green	242.813	4.589	246	255	255
Blue	247.308	4.747	250	227	255
GrayScale Intensity	243.194	4.671	246	225	255

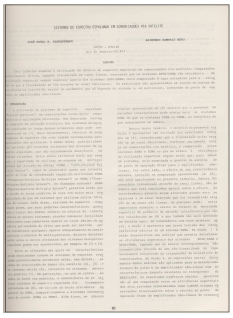


Figure 10.15: Historical Document 15

Table 10.15: Example Table (non-floating)

Label	Mean	Standard Deviation	Mode	Min	Max
Red	225.706	3.877	225	210	242
Green	219.162	3.862	219	205	233
Blue	206.987	4.248	207	192	224
GrayScale Intensity	219.685	3.668	219	206	234

References

Abburu, S.; Golla, S. B. Satellite Image Classification Methods and Techniques: A Review. *International Journal of Computer Applications*, v. 119, n. 8, p. 20–25, 2015.

Abramson, N. *Information Theory and Coding*. Mc Graw-Hill Book Company, 1963.

Academic, Microsoft. *Academic Social Networks*. http://academic.research.microsoft.com/, accessed: December the 3rd, 2013.

Academic, Microsoft. *Academic Social Networks*. http://academic.research.microsoft.com/, accessed: November the 15th, 2016.

Al-Hinnawi, A. R.; Daer, M. Assessment of bilateral filter on low NEX open MRI views. *SIViP*, 9–17, 2015.

Alaei, A.; Delalandre, M.; Girard, N. Logo Detection Using Painting Based Representation and Probability Features. *12th International Conference on Document Analysis and Recognition*, 1267–1271, 2013.

Anderson, T. W. *An Introduction to Multivariate Analysis*. 2. ed. error, 1984. v. 1.

Antonacopoulos, A.; Clausner, C.; Papadopoulos, C.; Pletschacher, S. ICDAR2013 Competition on Historical Newspaper Layout Analysis Ű HNLA2013. *12th International Conference on Document Analysis and Recognition*, 1454–1458, 2013.

ArnetMiner. *Academic Social Networks*. https://aminer.org, accessed: November the 18th, 2013.

ArnetMiner. *Academic Social Networks*. https://aminer.org, accessed: November the 15th, 2016.

Arruda, A. W. A.; Mello, C. A. B. Binarization of Degraded Document Images Based on Combination of Contrast Images. *14th International Conference on Frontiers in Handwriting Recognition*, 615–620, 2014.

Aurich, V.; Weule, J. Non-linear Gaussian Filters Performing Edge Preserving Diffusion. *Proceedings of the DAGM symposium*, 1995.

Avila, B. T.; Lins, R. D. A Fast Orientation and Skew Detection Algorithm for M. Document Images. *ACM DocEng*, 118–126, 2005.

Barata, R. B.; Goldbaum, M. A Profile of Researchers in Public Health with Productivity Grants from the Brazilian National Research Council. *CNPq*, v. 19, n. 6, p. 1863–1876, 2003.

Bataineh, B.; Abdullah, S. N. H. S.; Omar, K. An adaptive local binarization method for document images based on a novel thresholding method and dynamic windows. *Pattern Recognition Letters*, v. 32, p. 1805–1813, 2011.

Brocher, Jan. Qualitative and Quantitative Evaluation of Two New Histogram Limiting Binarization Algorithms. *International Journal of Image Processing*, v. 8, n. 2, p. 30–48, 2014.

Cavalcanti, G. D. C.; Silva, E. F. A.; Zanchettin, C.; Bezerra, B. L. D.; Doria, R. C.; Rabelo, J. C. B. A Heuristic Binarization Algorithm for Documents with Complex. *13th IEEE International Conference on Image Processing*, 389–392, 2006.

Chen, K.; Yin, F.; Liu, C. L. Hybrid Page Segmentation with Efficient Whitespace Rectangles Extraction and Grouping. *12th International Conference on Document Analysis and Recognition*, 958–962, 2013.

Costas, R.; Bordons, M. Is g-index better than h-index? An exploratory study at the individual level. *Scientometrics*, v. 77, n. 2, p. 267–288, 2008.

Crammer, J. S. *Logit Models from Economics and other Fields*. Cambridge University Press, 2003.

Dagar, A.; Archana; Nandal, D. High performance Computing Algorithm Applied in Floyd Steinberg Dithering. *International Journal of Computer Applications*, v. 43, n. 23, p. 11–13, 2012.

Dawoud, A.; Kamel, M. S. Iterative Model-Based Binarization Algorithm for Cheque Images. *International Journal on Document Analysis and Recognition*, v. 5, p. 28–38, 2002.

Dawoud, A.; Kamel, M. S. Iterative Multimodel Subimage Binarization for Handwritten Character Segmentation. *IEEE Transactions on Image Processing*, v. 13, n. 9, p. 1223–1230, 2004.

Diamantatos, P.; Kavallieratou, E.; Gil, P. G. Binarization: a Tool for Text Localization. *14th International Conference on Frontiers in Handwriting Recognition*, 649–654, 2014.

Diem, M.; Fiel, S.; Garz, A.; Keglevic, M.; Kleber, F.; Sablatnig, R. ICDAR2013 Competition on Handwritten Digit Recognition (HDRC 2013). *12th International Conference on Document Analysis and Recognition*, 1454–1459, 2013.

Egghe, L. Theory and practise of the g-index. *Scintometrics*, v. 69, n. 1, p. 131–152, 2006.

Egghe, L. An Improvement of the h-Index: The g-Index, 2007.

Elzobi, M.; Al-Hamadi, A.; Dings, L.; Elmezain, M.; Saeed, A. A Hidden Markov Model-based Approach with an Adaptive Threshold Model for Off-line Arabic Handwriting Recognition. *12th International Conference on Document Analysis and Recognition*, 945–949, 2013.

Floyd, R. W.; Steinberg, L. An Adaptative Algorithm for Spatial Greyscale. *Proceddings SID*, v. 17, n. 2, p. 75–77, 1976.

FUNDAJ. *Joaquim Nabuco Fundation.* http://www.fundaj.gov.br, accessed: february the 3rd, 2016.

Gaceb, D.; Lebourgeois, F.; Duong, J. Adaptative Smart-Binarization Method. *12th International Conference on Document Analysis and Recognition*, 118–122, 2013.

Gatos, B.; Pratikakis, I.; Perantonis, S. J. Adaptive Degraded document Image Binarization. *The Journal of the Pattern Recognition Society*, v. 39, p. 317–327, 2006.

Gatos, B.; Ntirogiannis, K.; Pratikakis, I. ICDAR 2009 Document Image Binarization Contest (DIBCO 2009). *10th International Conference on Document Analysis and Recognition*, 1375–1382, 2009.

Gupta, M. R.; Jacobson, N. P.; Garcia, E. K. OCR binarization and image preprocessing for searching historical documents. *The Journal of the Pattern Recognition Society*, v. 40, p. 389–397, 2007.

Hair, Joseph F. *Multivariate Data Analysis*. 7. ed. error, 2009. v. 1.

Hampson, G. *Especialista avalia mudanças na publicação de revistas científicas*. http://agencia.fapesp.br/especialista avalia mudancas na publicacao de revistas cientificas/24548/, accessed: January the 5th, 2017.

Hedjam, R.; Moghaddam, R. F.; Cheriet, M. A spatially adaptive statistical method for the binarization of historical manuscripts and degraded document images. *Pattern Recognition*, v. 44, n. 9, p. 2184–2196, 2011.

Helland, T. *Image Dithering*. http://www.tannerhelland.com/4660/dithering-eleven-algorithms-source-code/, december, 23, 2014.

Henault, D. R.; Moghaddam, R.F.; Cheriet, M. A local linear level set method for the binarization of degraded historical document images. *International Journal on Document Analysis and Recognition*, v. 15, p. 101–124, 2012.

Hirsch, J. E. An Index to Quantify an Individual's Scientific Research Output. *Physics Society*, September, 2005.

Howe, N. R. Document binarization with automatic parameter tuning. *International Journal on Document Analysis and Recognition*, v. 16, n. 3, p. 247–258, 2013.

Johannsen, G; Bille, J. A Threshold Selection Method Using Information Measure. *ICPR'82 - Proceeding 6th International Conference on Pattern Recognition*, 140–143, 1982.

Johnson, Richard A. ; Wichern, Dean W. *Applied Multivariate Statistical Analysis*. 6th. ed. Pretince Hall, 2007. v. 1.

Kapur, J. N.; Sahoo, P. K.; Wong, A. K. C. A New Method for Gray-Level Picture Thersholding Using the Entropy of the Histogram. *Computer Vision Graphics and Image Processing*, v. 29, p. 273–285, 1985.

Kasturi, R.; O'Gorman, L.; Govindaraju V. Document image analysis: A primer. *Sadhana*, 3–22, 2002.

Kavallieratou, E.; Antonopoulou, H. Cleaning and Enhancing Historical Document Images. *Intelligent Vision Systems*, 681–688, 2005.

Kittler, J.; Illingworth, J. Minimum Error Thresholding. *Pattern Recognition*, v. 19, n. 1, p. 41–47, 1986.

Kumar, G.; Bhatia, P. K. A Detailed Review of Feature Extraction in Image Processing Systems. *Fourth International Conference on Advanced Computing and Communication Technologies*, 5–12, 2014.

Labbé, C. Ike Antkare, One of the Great Stars in the Scientific Firmament. *International Society for Scientometrics and Informetrics Newsletter*, v. 6, n. 2, p. 48–52, 2010.

Lattes. *Academic Social Networks*. http://lattes.cnpq.br/, accessed: November the 20th, 2013.

Lattes. *Academic Social Networks*. http://lattes.cnpq.br/, accessed: November the 15th, 2016.

Lazaro, J.; Martin, J. L.; Arias, J.; Astarloa, A.; Cuadrado, C. Neuro semantic thresholding using OCR software for high precision OCR applications. *Image and Vision Computing*, v. 28, p. 571–578, 2010.

Leedham, G.; Varma S.; Patankar A.; Govindaraju V. Separating Text and Background in Degraded Document Images - A Comparison of Global Thresholding Techniques for Multi-Stage Thresholding. *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR02)*, 244–249, 2002.

Lelore, T.; Bouchara, F. Super-resolved binarization of text based on the FAIR algorithm. *2011 International Conference on Document Analysis and Recognition*, 839–843, 2011.

Lins, R. D.; Silva, G. F.; Simske S. J.; Fan J.; Shaw M.; Sa P.; Thielo M. Image Classifacation to Improve Printing Quality od Mixed-Typed Documents. *ICDAR 2009*, 1106–1110, 2009a.

Lins, R. D.; Silva, G. P.; Torreão G.; Alves N. F. Efficiently Generating Digital Libraries of Proceedings with The LiveMemory Platform. *The 7th International Telecommunications Symposium (ITS 2010)*, 2010a.

Lins, R. D.; Silva, G.; Silva J. M. Assessing Strategies to Remove Back-to-Front Interference in Color Documents. In: ResearchGate (Ed.), International Telecommunications Symposium, Septemberb, 2010.

Lins, R. D. A Taxonomy for Noise Detection in Images of Papers Documents-The Physical Noises. *Spring Verlag*, v. 4627, p. 844–854, 2009b.

Lins, R. D.; Silva, G.F. Removing Shade and Specular Noise by using Multiplr Images od Objects and Documents. *Springer Verlag*, 70–79, 2013.

Lins, R. D.; Silva, G. F. P; Formiga, A. A. HistDoc v. 2.0 Enhancing a Platform to Process Historical Documents. *Historical Document Imaging and Processing*, September, p. 169–176, 2011.

Lins, R. D. et al. An Enviromenment for Processing Images of Historical Documents. *Microproc. and Microprogramming*, 111–121, 1995.

Liu, Y.; Srihari, S. N. Document Image Binarization Based on Texture Features. *IEEE Transaction on Pattern Analysis and Machine Inteligence*, v. 19, n. 5, p. 540–544, 1997.

Lu, S.; Su, B.; Tan, C. L. Document image binarization using background estimation and stroke edges. *International Journal on Document Analysis and Recognition*, v. 13, p. 303–314, 2010.

Mello, C. A. B.; Sanchez, A.; Oliveira A. L. I. Image Thresholding of Historical Documents: Application to the Joaquim Nabuco's File. *Eva Vienna*, 115–122, 2006.

Mello, C. A. B.; Lins, R. D. Image segmentation of historical documents. *Visual 2000*, 2000.

Mello, C. A. B.; Lins, R. D. Generation of Images of Historical Documents by Composition. *Proceedings of the 2002 ACM symposium on Document engineering*, 127–133, 2002.

Memarsadeghi, N.; Mount, D. M.; Netanyahu N. S.; Moigne J. A Fast Implementation of the IsoData Clustering Algorithm. *International Journal of Computational Geometry and Applications*, 71–103, 2007.

Milyaev, S.; Barinova, O.; Novikova, T.; Kohli, P.; Lempitsky, V. Image binarization for end-to-end text understanding in natural images. *12th International Conference on Document Analysis and Recognition*, 128–132, 2013.

Moghaddam, R. F; Cheriet, M. AdOtsu: An adaptive and parameterless generalization of Otsu's method for document image binarization. *Pattern Recognition*, v. 45, p. 2419–2431, 2012.

Monte da Silva, J. M.; Lins, R. D.; Martins, F. M. J.; Wachenchauzer, R. A New and Efficient Algorithm to Binarize Document Images Removing Back-to-Front Interference. *Journal of Universal Computer Science*, v. 14, n. 2, p. 293–313, 2008.

Mukherjee, A.; Kanrar, S. Enhancement of Image Resolution by Binarization. *International Journal of Computer Applications*, v. 10, n. 10, p. 15–19, 2010.

Nafchi, H. Z.; Moghaddam, R. F.; Cheriet, M. Phase-based binarization of ancient document images: Model and applications. *IEEE Transactions on Image Processing*, 1–14, 2014.

Niblack, W. *An Introduction to Digital Image Processing*. Error, 1986.

Ntirogiannis, K.; Gatos, B.; Pratikakis, I. A combined approach for the binarization of handwritten document images. *Pattern Recognition Letters*, v. 35, p. 3–15, 2014a.

Ntirogiannis, K.; Gatos, B.; Pratikakis, I. ICFHR2014 Competition on Handwritten Document Image Binarization. *14th International Conference on Frontiers in Handwriting Recognition*, 809–813, 2014b.

of Science, Web. *Academic Social Networks*. https://webofknowledge.com, accessed: Novembr the 20th, 2013.

of Science, Web. *Academic Social Networks*. https://webofknowledge.com, accessed: Novembr the 15th, 2016.

Oha, H.; Limb, K.; Chienc S. An improved binarization algorithm based on a water flow model for document image with inhomogeneous background. *Pattern Recognition*, 2612–2625, 2005.

Omohundro, Stephen M. Floyd-Steinberg Dithering. *International Computer Science Institute*, October, 1990.

Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transaction on Systems, Man and Cybernetics*, v. SMC-9, n. 1, p. 62–66, 1979.

Pai, Y.; Cheng, Y.; Ruan, S. Adaptive thresholding algorithm: Efficient computation technique based on intelligent block detection for degraded document images. *Patern Recognition*, v. 43, p. 3177–3187, 2010.

Papandreou, A.; Gatos, B.; Louloudis, G.; Stamatopoulos, N. ICDAR2013 Document Image Skew Estimation Contest (DISECŠ13). *12th International Conference on Document Analysis and Recognition*, 1476–1480, 2013.

Paris, S.; Durand, F. A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach. *International Journal of Computer Vision*, v. 1, n. 81, p. 24–52, 2009.

Parkera, J.; Friedera, O.; Frieder, G. Robust Binarization of Degraded Document Images Using Heuristics. *Document Recognition and Retrieval XXI - SPIE-IS and T Electronic Imaging*, v. 9021, p. 1–12, 2014.

Pratikakis, I.; Gatos, B.; Ntirogiannis, K. H-DIBCO 2010 Ű Handwritten Document Image Binarization Competition. *12th International Conference on Frontiers in Handwriting Recognition*, 727–732, 2010.

Pratikakis, I.; Gatos, B.; Ntirogiannis, K. ICDAR 2011 Document Image Binarization Contest (DIBCO 2011). *2011 International Conference on Document Analysis and Recognition*, 1506–1510, 2011.

Pratikakis, I.; Gatos, B.; Ntirogiannis, K. ICDAR 2013 Document Image Binarization Contest (DIBCO 2013). *12th International Conference on Document Analysis and Recognition*, 1471–1476, 2013.

Pun, T. Entropic Thresholding, A New Approach. *Computer Vision Graphics and Image Processing*, 210–239, 1981.

Rabeux, V.; Journet, N.; Vialard, A.; Domenger, J. Quality evaluation of ancient digitized documents for binarization prediction. *12th International Conference on Document Analysis and Recognition*, 113–117, 2013.

Ramirez-Ortegon, M. A.; Ramirez-Ramirez, L. L.; Margner, V.; Messaoud, I. B.; Cuevas, E.; Rojas, R. An analysis of the transition proportion for binarization in handwritten historical documents. *Pattern Recognition*, v. 47, p. 2635–2651, 2014.

Richards, J. A. *Remote Sensing Digital Image Analysis: An Introduction*. second. ed., 1993.

Sahoo, P. K.; Soltani, S.; Wong A. K. C. A Survey of Thresholding Techniques. *Computer Vision, Graphics, and Image Processing*, 233–260, 1988.

Sauvola, J.; Pietikainem, M. Adaptive document image binarization. *The Journal of the Pattern Recognition Society*, v. 33, p. 225–236, 2000.

Scholar, Google. *Academic Social Networks*. https://scholar.google.com.br, accessed: December the 2nd, 2013.

Scholar, Google. *Academic Social Networks*. https://scholar.google.com.br, accessed: November the 15th, 2016.

Scopus. *Academic Social Networks*. https://www.elsevier.com/solutions/scopus, accessed: November the 20th, 2013.

Scopus. *Academic Social Networks*. https://www.elsevier.com/solutions/scopus, accessed: November the 15th, 2016.

Seki, M.; Asano, E.; Yasue, T.; Nagayoshi, H.; Shinjo, H.; Nagasaki, T. Color Drop-out Binarization Method for Document Images with Color Shift. *12th International Conference on Document Analysis and Recognition*, 123–127, 2013.

Sezgin, M.; Sankur, B. A Survey over Image Thresholding Techniques and Quantitative Performance Evaluation. *Journal of Electronc Imaging*, v. 1, n. 13, p. 146–165, 2004.

Shah, L.; Patel, R.; Patel, S.; Maniar, J. Handwritten Character Recognition using Radial Histogram. *International Journal of Research in Advent Technology*, v. 2, n. 4, p. 24–28, 2014.

Shaikh, S. H.; Maiti, A. K.; Chaki, N. A new image binarization method using iterative partitioning. *Machine Vision and Applications*, v. 24, n. 2, p. 337–350, 2013.

Sharma, G. Show-trough cancellation in scans of duplex printed documents. *IEEE Transaction Image Processing*, v. 10, n. 5, p. 736–754, 2001.

Shi, J.; Ray, N.; Zhang, H. Shape based local thresholding for binarization of document images. *Pattern Recognition Letters*, v. 33, p. 24–32, 2012.

Silva, G. P.; Lin, R. D. PhotoDoc: A Toolbox for Processing Document Images Acquired Using Portable Digital Cameras. *CBDAR 2007*, 107–114, 2007.

Silva, J. M. M.; Lins, R.D.; Rocha V. C. Binarizing and Filtering Historical Documents with Back-to-Front Interference. *Proceedings of the 2006 ACM symposium on Applied Computing*, 853–858, 2006.

Simske, S. J. Low Resolution Photo/drawing Classification: metrics, method and archiving optimization. *IEEE ICIP*, 534–537, 2005.

Sokratis, V.; Kavallieratou, E.; Paredes, R.; Sotiropoulos, K. *Learning Structure and Schemas from Documents*. Springer-Verlag Berlin Heidelberg, 2011. Cap. A Hybrid Binarization Technique for Document Images, p. 165–179.

Su, B.; Lu, S.; Tan, C. L. Robust Document Image Binarization Technique for Degraded Document Images. *IEEE Transactions on Image Processing*, v. 22, n. 4, p. 1408–1417, 2013.

Sumathi, C. P.; Santhanam, T.; Devi, G. G. A Survey on Various Approaches of Text. *International Journal of Computer Science and Engineering Survey*, v. 3, n. 4, p. 27–42, 2012.

Tan, C. L.; Cao, R.; Shen P. Restoration of archival documents using a wavelet technique. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, v. 24, n. 10, p. 1399–1404, 2002.

Tomasi, C.; Manduchi, R. Bilateral Filtering for Gray and Color Images. *Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India*, 1998.

Trier, O. D.; Jain, A. K. Goal-directed evaluation of binarization methods. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, v. 17, n. 12, p. 1191–1201, 1995.

Valizadeh, M.; Kabir, E. Binarization of degraded document image based on feature space partitioning and classification. *IJDAR*, v. 15, p. 57–69, 2012.

Wang, Q.; Tan, C. L. Matching of Double-Sided Document Images to Remove Interference. *IEEE CVPR 2001*, 1084–1089, 2001.

Weszka, J. S.; Rosenfeld, A. Histogram Modification for Threshold Selection. *IEEE Transaction on Systems, Man and Cybernetics*, v. SMC-9, n. 1, p. 38–52, 1979.

Wu, L. U.; Songde, A.; Haqing L. U. An Effective Entropic Thresholding for Ultrasonic Imaging. *International Conference Pattern Recognition*, 1522–1524, 1998.

Yadav, J.; Sharma, M. A Review of K-mean Algorithm. *International Journal of Engineering Trends and Technology (IJETT)*, v. 4, n. 7, p. 2972–2976, 2013.

Yen, J. C.; Chang, F. J.; Chang S. A New Criterion for Automatic Multilevel Thresholding. *IEEE Transaction Image Process IP-4*, 370–378, 1995.