Technic Report- Generating functions/trees for evaluating optimal binarizations

1st Crina-Andreea Gherase
Project Manager
National University of Science and Technology Politehnica Bucharest

2nd Nicolae-Cosmin Cornea
Team Leader
National University of Science and Technology Politehnica Bucharest

3rd Ștefan Costin Manta
Software Developer
National University of Science and Technology Politehnica Bucharest

4th Mara-Ilinca Mocanu
Software Developer
National University of Science and Technology Politehnica Bucharest

5th Paul-Iulian Andrei
Tester
National University of Science and Technology Politehnica Bucharest

***Abstract*—This report outlines a project that focuses on improving document digitization through enhanced binarization. It introduces two solutions, one for global and another for local binarization, aiming to intelligently combine algorithm outputs.**

***Index Terms*—Global and Local Binarization, Image Processing, F-measure, Document Digitization.**

I. Introduction

Historical documents suffer from various degradations due to ageing, extended use, several attempts of acquisition and environmental conditions [1]– [4]. The main artefacts encountered in historical documents are shadows, non-uniform illumination, smear, strain, bleed-through and faint characters. Those artefacts are problematic for document image analysis methods which assume smooth background and uniform quality of writing [1]. In handwritten documents, the writer may use different amount of ink and pressure and generate characters of different intensity or thickness, as well as faint characters. The same writer may write in different ways even within the same document. Similar problems, such as faint characters and non-uniform appearance of characters of the same font, are also encountered in historical machine-printed documents[5].

In image processing and document analysis, global binarization is a basic procedure that turns a grayscale image into a binary image in which pixels are categorised as background or foreground depending on a global threshold value. Global binarization aims to simplify the image and improve its contrast, which will facilitate the extraction of significant information for further analysis.

Global binarization is essential for addressing a number of issues related to document degradation in the context of historical document research. As was previously indicated, artefacts including shadows, uneven lighting, smearing, strain, bleed-through, and pale lettering are common in historical texts. Conventional document analysis techniques may find it challenging to read the content effectively in light of these artefacts.

By creating a threshold that distinguishes the foreground (text or pertinent information) from the background in a way that is applicable to the entire document, global binarization approaches seek to address these issues. Choosing a suitable threshold is an essential stage since it affects the precision of the processing stages that come after. Various approaches, including Otsu's method or adaptive thresholding, can be used, depending on the type of degradation and the document's properties.

Global binarization techniques must be strong enough to manage such complications when dealing with historical handwritten texts, where variances in ink strength, pressure, and writing style are frequent. The goal is to produce a binary representation that faithfully represents the document's content, enabling more accurate and efficient analysis for tasks like optical character recognition (OCR) or text extraction.

Overall, global binarization is a key preprocessing step in the analysis of historical documents, serving as the foundation for subsequent tasks aimed at restoring, preserving, and understanding valuable historical materials. Developing effective global binarization methods is essential for ensuring the success of document image analysis techniques applied to diverse and degraded document collections.

In contrast to global binarization, local binarization, also known as adaptive thresholding, recognizes that the optimal threshold value can vary across different regions of an image. This technique divides the image into smaller, localized regions and applies a threshold individually to each of these regions. The adaptive nature of local binarization allows it to better handle variations in intensity, background, and degradation within different parts of a document.

In the context of historical document analysis, local binarization becomes particularly valuable when dealing with documents that exhibit uneven illumination, faded ink, or variations in writing characteristics. Traditional global thresholding methods may struggle to adapt to these local variations, leading to inaccurate segmentation and subsequent analysis.

Local binarization algorithms, such as Niblack, Sauvola, or Nick, calculate the threshold for each local region based on the statistical properties of the pixels within that region. These methods are effective in preserving fine details and ensuring accurate segmentation, especially when applied to documents with complex backgrounds or intricate writing styles.

The advantage of local binarization lies in its ability to address the challenges posed by diverse document conditions within a single image. By adjusting the threshold dynamically based on local characteristics, it helps maintain the integrity of the document's content and enhances the accuracy of subsequent analysis tasks, such as OCR or feature extraction.

In document image processing, combining global and local binarization algorithms in a multi-stage approach is a popular strategy. By combining the best features of both approaches, this hybrid strategy optimises the binarization process for old documents with a variety of degradation patterns and heterogeneous content.

To sum up, local binarization is an effective preprocessing tool that can be used to adapt to a variety of specific localised issues that aged and damaged documents bring. It is an essential part of the pipeline for historical document analysis. When included into workflows for document image analysis, it produces more accurate and significant outcomes for historical material preservation, restoration, and interpretation.

II. Proposed Solution

A. Global Binarization

Our Java application has as the main technical solution the class "GenerateTrees" that contains all the logic for building trees. The GenerateTrees class represents a solution for global binarization by dynamically creating and generating trees with random mathematical functions. The class utilizes the Operations class, which contains various mathematical functions with one, two, or multiple arguments.

The main functionality of the solution lies in the generateTree method, which creates a binary tree structure with 10 leaves. The leaves are randomly selected from a provided set of test values. The tree is constructed by recursively applying random mathematical functions to the selected values until a single value remains.

While the solution may seem complex due to the use of reflection for dynamic method invocation, it provides a flexible and extensible framework for incorporating a variety of mathematical operations into the global binarization process. The resulting trees can be used for further analysis or experimentation in the context of the problem domain.

After careful evaluation, we chose to generate 100 trees and save them to determine the best 10 ones.

B. Local Binarization

For local binarization, the top 10 global trees were used to generate local trees, which were then evaluated on the test dataset.

C. Running the generated trees

The RunTrees class functions as the orchestrator for the execution of the global binarization process. Its generateTrees method takes charge of initiating the simultaneous creation of 100 trees by utilizing the GenerateTree class in a parallel manner. Subsequently, the runTrees method orchestrates the concurrent execution of each generated tree, employing all dataset values. This process results in the production of an individual result file for every tree.

On the other hand, the RunTree class encapsulates the responsibilities associated with executing a specific tree on the entire dataset. The applyFunction method dynamically triggers mathematical functions, adapting to the provided function and the number of arguments. In conjunction, the calculateTreeValue method reads the tree structure generated from a file, applies the corresponding functions to the dataset values, and computes the conclusive result. The run method takes charge of executing the tree calculation for a designated file index and dataset row, subsequently inscribing the outcomes to a file.

III. Architecture

For our application our team chose to use a layered architecture which consists of the following layers:

A. Presentation Layer

Main Class: Responsible for starting and initializing the application. Contains the main method to kick off the application.

B. Business Logic Layer

GenerateTrees Class: Contains the logic for generating mathematical trees based on input datasets. Represents the core business logic of the application. Operations Class: Contains various mathematical operations used in tree generation.

C. Data Access Layer

CsvReader Class: Acts as the interface between the application and external CSV data sources. Reads data from CSV files, parsing it into a structure used for testing trees. PixelReader Class: Similar to CsvReader but specifically designed for reading pixel-related data from CSV files.

D. Application Layer

RunTrees Class: Orchestrates the execution of tree-generation and testing tasks. Utilizes parallel processing for tree generation and runs trees on pixel values in parallel. Testing Class: Performs specific tests and calculations using the generated trees and input datasets.

E. Interaction Flow

The Main class initiates the application, creating instances of CsvReader and RunTrees. After that, the RunTrees class coordinates the parallel execution of tree-generation tasks and interacts with the GenerateTrees class and global datasets. The GenerateTrees class implements the core logic for generating mathematical trees and utilizes the Operations class for mathematical computations. The CsvReader and PixelReader classes handle reading data from CSV files, providing data structures for tree generation and testing. The Operations class offers reusable mathematical operations for tree generation.The Testing class performs specific tests and calculations using the generated trees and datasets.

F. Relationships

The Main class serves as the entry point, initializing the application and triggering tree generation and testing processes. The RunTrees class manages the parallel execution of tree-generation tasks and interacts with the GenerateTrees class. The GenerateTrees class contains the core logic for tree generation and relies on the Operations class for mathematical computations. The CsvReader and PixelReader classes handle data access by reading information from CSV files. The Operations class provides a set of reusable mathematical operations. The Testing class performs specific tests and calculations using the generated trees and datasets. This architecture provides a clear separation of concerns and promotes maintainability and extensibility. It also allows foreasier testing of individual components and encourages good coding practices. Depending on the specific requirements and complexity of the application, the architecture can be adjusted or expanded.

IV. Results

A. Global Binarization

Out of the 100 generated trees, these had the best results:
TreeNr36: Average F-Measure - 69.76
TreeNr26: Average F-Measure - 69.50
TreeNr17: Average F-Measure - 67.75
TreeNr91: Average F-Measure - 65.42
TreeNr100: Average F-Measure - 63.14
TreeNr72: Average F-Measure - 62.77
TreeNr43: Average F-Measure - 61.63
TreeNr6: Average F-Measure - 60.76
TreeNr45: Average F-Measure - 60.33
TreeNr89: Average F-Measure - 59.10

B. Local Binarization

Out of our 10 best generated trees, these had the best results:
TreeNr100: F-Measure - 0.20
TreeNr43: F-Measure - 0.18
TreeNr91: F-Measure - 0.18

V. Conclusions

In conclusion, our project addresses the critical issue of improving document digitization, particularly in the context of historical documents with various degradations. We proposed two innovative solutions: global binarization and local binarization, each tailored to handle specific challenges inherent in historical document analysis.

References

[1] A. Antonacopoulos and D. Karatzas, ”Semantics-based contentextraction in typewritten historical documents”, Proc. Int. Conf. Document Anal. Recognit., pp. 48-53, 2005.

[2] B. Gatos, I. Pratikakis and S. J. Perantonis, ”Adaptive degraded document image binarization”, Pattern Recognit., vol. 39, no. 3, pp. 317-327, Mar. 2006.

[3] A. Antonacopoulos, D. Karatzas, H. Krawczyk and B. Wiszniewski, ”The lifecycle of a digitalhistorical document: Structure and content”, Proc. ACM Symp. Document Eng., pp. 147-154, 2004.

[4] A. Antonacopoulos and D. Karatzas, ”Document image analysis for world war II personalrecords”, Proc. Int. Workshop Document Image Anal. Libraries, pp. 336-341, 2004.

[5] K. Ntirogiannis, B. Gatos and I. Pratikakis, ”Performance Evaluation Methodology for Historical Document Image Binarization,” in IEEE Transactions on Image Processing, vol. 22, no. 2, pp. 595-609, Feb. 2013, doi: 10.1109/TIP.2012.2219550.