Generating functions/trees for evaluating optimal binarizations

TECHNICAL REPORT

1st Asmaa Alaghbari	2nd Mihai Ilinca	3rd Ioana Rusu	4th Andreea-Maria Piciu	5th Cerasela Enus
Team Leader	*Developer*	*Developer*	*Project Manager*	*Tester*
343C5	*343C5*	*342C3*	*341C5*	*343C5*

***Abstract*—This project proposes a new image binarization method that combines global and local thresholding. The proposed method was evaluated on a variety of benchmark datasets and outperformed existing methods in terms of accuracy, recall, and F-measure.**

***Index Terms*—Image binarization, Global thresholding, Local thresholding, Decision Trees, Document digitization.**

I. Introduction

Image binarization is the process of converting a grayscale image to a binary image, where each pixel is assigned a value of either black or white. Binarization is a fundamental image processing task that is used in a variety of applications, such as text recognition, document analysis, and medical imaging. There are a variety of different binarization methods available, each with its own advantages and disadvantages. Global thresholding methods use a single threshold value to binarize the entire image. This approach is simple and efficient, but it can be sensitive to noise in the image.

Local thresholding methods use different threshold values for different parts of the image. This approach is more robust to noise than global thresholding, but it is more complex and computationally expensive.

II. Solution

Our solution is a sophisticated image binarization system designed to outperform existing global and local thresholding methods. The decision trees are constructed dynamically by analyzing the dataset consisting of various thresholding outputs, each corresponding to different binarization algorithms applied to a set of images. This dataset is preprocessed and normalized, ensuring that the inputs to our system are consistent and within a defined operational range.

In the context of the code implementation, the 'functions.py' module encapsulates a suite of mathematical functions that serve as potential operations within the nodes of our trees. These include statistical measures such as mean, variance, and standard deviation, as well as more complex operations that can be applied to the thresholding data.

The 'main.py' script acts as the orchestrator of our system, handling the reading and parsing of CSV files, initiating the tree generation process, and managing the evaluation of the generated trees. This script has been designed to facilitate easy extension and integration with additional data sources and algorithmic modules.

Through the execution of 'main.py', the system generates a variety of decision trees, each representing a unique combination of thresholding strategies. These trees are then evaluated based on their F-measure performance against a set of ground truth images.

III. Architecture

Our system's architecture is modular and extensible, comprising several distinct components:

- Data Reader: Implemented in 'main.py', this component is responsible for ingesting and normalizing data from CSV files. It allows for flexible data source integration and is designed to handle large volumes of data efficiently.
- Tree Generator: As defined in 'tree.py', this component uses a stochastic process to generate decision trees. It randomly selects from a pool of mathematical functions and binarization thresholds to construct trees with varying structures and operational logic.
- Evaluator: This module, integrated within 'main.py', calculates the F-measure of each tree by applying the generated thresholds to the data and comparing the output to the ground truth.
- Logger: A logging mechanism is built into the system to record the performance and characteristics of each tree, aiding in the analysis and iterative improvement of the system.

The architecture also features parallel processing capabilities, allowing for multiple trees to be generated and evaluated concurrently, significantly speeding up the experimentation and optimization phases.

IV. Intermediate Results

The main.py program uses the generated trees to evaluate the performance of binarizations on input data. The F-measure is used to quantify how well the optimal binarizations generated by the trees match the provided references. The average F-measure for all lines provides an overall estimate of the performance of the binarizations.

- Variability of Results: Because the trees are randomly generated, results may vary between different runs of the program. Variability may reflect the fact that, under certain circumstances, certain functions or values may lead to better binarizations than others.
- Possible Problems: If input data and references are not available, it is difficult to assess whether the binarizations are truly optimal or not. The use of representative datasets and appropriate references is crucial to ensure that the assessment is relevant and understandable.
- Parameter Optimization: If the performance of the binarizations is not satisfactory, one possibility would be to adjust the tree generation parameters or change the way functions and values are randomly chosen. Deeper exploration of function types and tree structure can provide insights into more effective binarization strategies.
  The intermediate results of our project have been highly encouraging. We have conducted extensive tests across multiple datasets, with our system achieving consistently higher F-measure scores compared to baseline thresholding methods.

V. Intermediate Conclusions

The intermediate conclusions of our work draw from the results obtained from the iterative design, implementation, and testing of our image binarization system. Key insights include:

- **Algorithmic Efficacy**: The integration of various binarization algorithms into a tree structure has proven effective. By not relying on a single global threshold or localized thresholds alone, our system adapts to the nuances of different image types and contents.
- **System Robustness**: The robustness of the system against noise and varying image qualities has been validated. This robustness is primarily attributed to the diverse mathematical operations that consider both global image characteristics and local nuances, as implemented in the ‘functions.py‘ module.
- **Scalability**: The system’s architecture, as laid out in ‘tree.py‘ and ‘main.py‘, supports scalability. It has handled extensive datasets with efficiency, and the modular design allows for easy expansion to include more complex algorithms or larger datasets.
- **Performance Metrics**: The decision trees have achieved an improvement in precision and recall metrics across the board, indicating a significant advancement over traditional thresholding techniques. This has been quantitatively supported by our F-measure evaluations.
- **Parallelization**: The successful implementation of parallel processing has not only met but exceeded our non-functional performance targets, highlighting the system’s capacity for high-throughput image processing.
- **User Experience**: While the current system operates via command line, the ease of use and the speed of execution suggest that a future graphical user interface could be both viable and beneficial for expanding the system’s user base.

These findings lead us to a positive outlook on the future development phases of the project. The next steps include refining the tree generation algorithm to improve decision-making further, exploring the integration of machine learning techniques to adaptively learn from the datasets, and implementing a graphical user interface to increase accessibility for users.

Additionally, the conclusions we have drawn underscore the potential impact of our solution in fields where image processing plays a critical role, such as digital archiving, medical diagnostics, and automated surveillance systems.

References

[1] CS Open CourseWare. (2023). Proiect [CS Open CourseWare]. Retrieved from https://ocw.cs.pub.ro/courses/mps/proiect

[2] Gonzalez, R.C., & Woods, R.E. (2008). Digital Image Processing (3rd Edition).

[3] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62-66.

[4] Niblack, W. (1985). An introduction to Digital Image Processing.