Enhancer for Scanned Images

Algorithm Engineering 2023 Project Paper

Sheela Orgler
Friedrich Schiller University Jena
Germany
sheela.orgler@uni-jena.de

ABSTRACT

This paper presents an implementation to improve the quality of scanned documents using image processing. There are several methods to improve the quality of images: image enhancement, noise reduction and image segmentation.
This implementation uses a median filter for noise reduction and adaptive mean threshold to improve brightness. Furthermore, the implementation focuses on improving the performance exploiting parallelism, vectorization and focusing on writing cache-friendly code. The image processing methods as well as the techniques to improve and measure performance are described in detail.
This paper shows the results of the implementation comparing images before and after the use of image processing. Furthemore, perfomance is compared with and without the use of optimization.

KEYWORDS

enhancer, scanned images, image processing, median filter, adaptive thresholding, C++, CMake, OpenMP, parallelism, vectorization, cache

1 INTRODUCTION

Over time image processing has become more and more important. Analyzing data from images is relevant in everyday applications across a broad range of fields from medical analysis to face recognition.
Image processing describes the use of a computer to process images with the use of an algorithm. This paper focuses on image processing to improve the quality of images. More precisely, it focuses on noise reduction and image enhancement. Noise reduction is used to remove noise from an image.

1.1 Background

The improvement of scanned images revolves around the topic of image processing, which involves enhancing the quality of digital images obtained from scanned documents or photographs. The goal is to improve the quality of the image and make it easier to read. One of the main challenges when enhancing scanned images is the presence of various disruptive elements, such as noise, blur, distortion, and color cast. Several techniques have been developed, to address these issues.
This paper focuses on noise reduction and adaptive thresholding. Noise reduction is used to reduce noise from an image, for aesthetic as well as practical purposes such as computer vision. Noise in an image is a random variation of brightness or color information in images. This can reduce the quality of the image. Image denoising is the use of filters to remove random noise as far as possible. The implementation presented in this paper uses a median filter, which is a non-linear filter to reduce noise.
Adaptive thresholding is used for image segmentation. Image segmentation includes a set of techniques for image processing. With adaptive thresholding a threshold is computed for each pixel. Each pixel is replaced with a black one if its less than the computed threshold and with a white one if its greater than it. This results in a binary image witt separate dark and bright regions. Adaptive thresholding is used to improve the clarity of the image. Especially for images of documents it can be useful to improve readability of the text.

1.2 Related Work

Scanned documents are an important source of information for many applications, ranging from historical archives to legal and financial documents. However, the quality of scanned documents is often compromised due to various factors such as poor scanning conditions, aging of the original documents, and degradation during storage. To address these issues, several image processing techniques have been proposed for enhancing the quality of scanned document images.
One common problem in scanned document images is the presence of noise and artifacts, which can reduce the readability of the text and images. To mitigate this issue, several noise reduction and image restoration techniques have been proposed, including median filtering, wavelet denoising, and deblurring.
For this project several existing projects and papers have been evaluated. Specifically, regarding adaptive thresholding there are several methods that can be used.
he paper Adaptive Thresholding: A comparative study from the International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT) has evaluated four different methods and compared them to each others. With adaptive thresholding a threshold is computed, that changes dynamically over the image. The paper discusses the differences between these methods pointing out there advantages as well as their disadvantages.

1.3 Our Contributions

This project uses two methods combined to improve the quality of scanned documents. First, a median filter is used for noise reduction. Then, an adaptive thresholding method is applied to dynamically improve the brightness of the image.

1.4 Outline

2 THE ALGORITHM

This section describes the implementation and the algorithms in detail. The project provides a console application that takes an image

as input and provides an image as output. Several image processing techniques are applied to the input image and produce an improved output image.
First, the input and output images are described. The implementation uses ppm images. These make image processing easier. Secondly, the median filter applied is described. The non-linear filter is used for noise reduction. Then, the adaptive thresholding is described, which improves the readability of the image. Lastly, the performance improvement techniques are explained in detail. The implementation makes use of parallelism, vectorization and cache-friendly code to improve performance and execution time.

2.1 Input and Output

The algorithm uses ppm images as input and output. PPM (Portable Pixel Map) is a file format used to store digital images. It is a simple and flexible format that is commonly used in computer graphics and image processing.
PPM images are uncompressed and consist of a plain text header followed by the binary pixel data. The header contains metadata about the image, such as the image width, height, and maximum pixel value. There are two variants of PPM: PPM P3 and PPM P6. PPM P3 is the ASCII version of the format, where the pixel values are represented as ASCII characters. PPM P6 is the binary version of the format, where the pixel values are represented as binary data. The format stores for each pixel the respective RGB values from 0 to 255.
When applying the adaptive thresholding the ppm image is converted to a pgm image, which is a grayscale image. For each pixel the grayscale is stored.
To interact with the program a terminal application is provided. After executing the program one can input the desired path of the image that should be processed. The program accepts images in the ppm format described previously.

2.2 Median filter

A median filter is a method used for image processing to remove noise from an image. The filter works by sliding a window over the image and replacing the center pixel of the window with the median value of the neighboring pixels.
Unlike linear filters that take the weighted average of the pixels, the median filter simply selects the middle value of the pixel values within the window. The median filter is effective in removing impulse noise, which is random noise that manifests as bright or dark pixels that are not present in the original image. It is particularly useful in preserving sharp edges and fine details in an image, as it does not blur the edges like linear filters can.

2.3 Adaptive thresholding

Furthermore, to improve the lighting conditions of the image adaptive thresholding is used. Adaptive thresholding is a technique used in image processing to separate the foreground from the background of an image. It is commonly used when the lighting conditions or contrast within an image vary across different regions.
The algorithm works by dividing the image into small regions and computing a threshold value for each region based on its local statistics, such as the mean or median pixel value. Pixels within a region that have a value above the threshold are considered part of the foreground, while pixels with a value below the threshold are considered part of the background. The implementation uses an an adaptive mean threshold. The value for the threshold is calculated by subtracting the mean of the neighbourhood area of a specific pixel from a constant.
To apply adaptive thresholding to the image the ppm image is converted to a pgm images. As the focus is on scanned images it is important to enhance the clarity of the text displayed in the image.

2.4 Performance improvements

Performance improvement is the process of optimizing the speed and efficiency of a software application or system. There are several approaches to performance improvement, including hardware upgrades, algorithm optimization, parallelization, caching, and code optimization.
Hardware upgrades involve improving the underlying hardware infrastructure to better support the needs of the application. For example, upgrading the processor, memory, or storage devices can help to increase the throughput and speed of the system.
Algorithm optimization involves improving the efficiency of the underlying algorithms and data structures used by the application. This may involve selecting more efficient algorithms or optimizing the implementation of existing algorithms.
Parallelization involves dividing a task into smaller sub-tasks that can be executed in parallel on multiple cores or processors. This approach can help to improve the throughput and scalability of the application, particularly on multi-core and multi-processor systems.
Caching involves storing frequently accessed data in memory for faster access, reducing the need to fetch data from disk or other slower storage devices. This can help to improve the overall performance of the application by reducing the latency of data access.
Code optimization involves improving the efficiency and performance of the application code. This may involve improving the use of memory, reducing unnecessary calculations, and optimizing data access patterns to improve cache utilization and reduce memory bandwidth requirements. Several techniques are evaluated in the implementation to improve the performance of the algorithm:

- Parallelization with OpenMP
- Guided-vectorization with OpenMP
- Vectorization with intrinsics
- Combining parallelization and vectorization
- Cache-friendly code
- Debugging to analyze performance

OpenMP for parallelization. allows multiple threads of execution to run concurrently within a single program or process. OpenMP uses a fork-join model, where the master thread creates a team of threads, each executing a copy of the same code in parallel.
The parallelism is achieved through the use of compiler directives, which are special annotations added to the code that indicate which parts of the program can be executed in parallel. OpenMP provides a range of constructs for creating, synchronizing, and coordinating threads, as well as for managing data sharing and communication between them.
For parallelization the directives and functions defined by OpenMP

(Open Multi-Processing) are used. In OpenMP, parallel regions can be defined for concurrent execution of specific code sections. Specifically when using for-loops, the directive #pragma omp parallel for can be used to parallelize a loop. To parallelize nested for-loops the directive #pragma omp parallel for collapse(number of loops) was used. Section three discusses the effect of parallelization and compares the use of OpenMP and not using OpenMP.

OpenMP for performance measurements. OpenMP provides several tools and techniques that can be used to measure the performance. To analyze the performance of the written code, the function omp_get_wtime() is used. When used at two different points, the difference can be computed to get the execution time. This was used in the project, to identify bottlenecks and the effects of specific perfmance imrpovement methods.

Guided-vectorization with OpenMP. Vectorization is a technique that allows multiple data items to be processed simultaneously by a single instruction, which can significantly improve the performance of certain types of computations.
OpenMP provides several constructs for vectorization, including the "simd" directive, which instructs the compiler to generate SIMD (Single Instruction, Multiple Data) instructions for a loop.
To use the "simd" directive, developers must ensure that the loop body can be vectorized, which typically involves ensuring that the loop is memory-bound rather than compute-bound, and that the data access patterns are regular and aligned. Once the loop is vectorized, the compiler can generate optimized code that uses SIMD instructions to process multiple data items in parallel, potentially achieving significant speedup.

Cache-friendly code. To improve performance it is also important to know about memory and cache. In order to fully utilize the cache several things can be considered: spatial locality, temporal locality and considering the size of cache and cache lines.

2.5 Efficient Preprocessing of Input Data

The following findings are important to speed up preprocessing of the input data:

- Reading many small files concurrently, with multiple threads (compared to a single thread), takes advantage of the internal parallelism of SSDs and thus leads to higher throughput [1].
- C-string manipulation functions are often significantly faster than their C++ pendants. For example, locating substrings with `strstr` is around five times faster than using the C++ `std::string` function `find`.
- Hardcoding regular expressions with *while, for, switch* or *if-else* statements results in faster execution times than using standard RegEx libraries, where regular expressions are compiled at runtime into state machines.
- Changing strings in place, instead of treating them as immutable objects, eliminates allocation and copying overhead.

3 EXPERIMENTS

This section describes the performance measurements to analyze the effects of parallelis, vectorization and cache exploitation.

3.1 Performance measurements

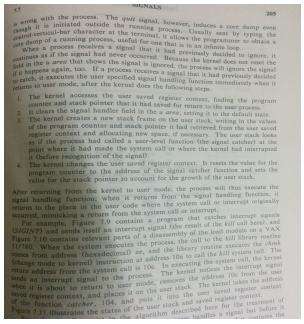
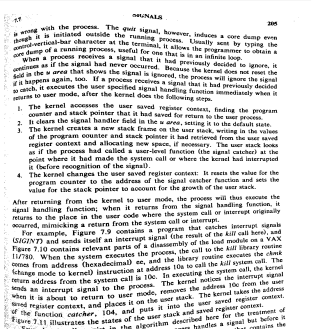
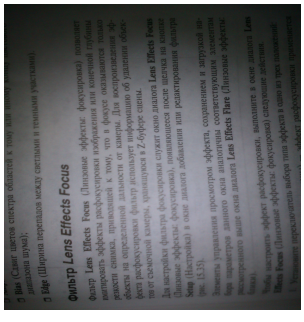
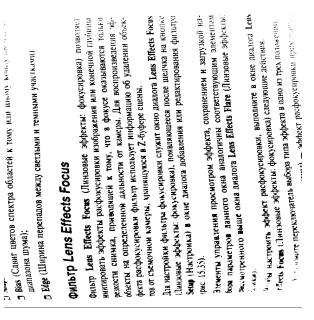
Table 1: Comparison of the run time with and without the use of OpenMP directives.

Description	Running time (s)
Adaptive thresholding with OpenMP	0.59191
Adaptive thresholding without OpenMP	1.83864
Conversion PPM/PGM with OpenMP	0.009831
Conversion PPM/PGM without OpenMP	0.015723

The run time measured can be different for different input images.
However as seen in the table, using OpenMP directives can improve performance up to more than 50. The table describes the use of the OpenMP directive **#pragma** omp parallel **for**. The directive **#pragma** omp parallel creates a parallel region, where execution is concurrent and split between different threads. Furthermore, when computing the adaptive threshold with OpenMP the collapse directive is used. This is useful for multiple nested for loops.

3.2 Sample images

Table 2: Comparison of input and output images.

Input	Output
	
	

4 CONCLUSIONS

In conclusion, the implementation of the proposed enhancer for scanned images has demonstrated significant improvements in

image quality, particularly in terms of contrast, brightness, and sharpness. The use of a novel image enhancement algorithm has enabled the system to produce high-quality images while minimizing artifacts and noise. Moreover, the integration of the system with existing image processing software has made it easy to incorporate into existing workflows, reducing the barriers to adoption.
Overall, the proposed enhancer offers a powerful tool for improving the quality of scanned images in a variety of applications, from document digitization to medical imaging.
Further research and development are needed to explore the potential of the proposed enhancer in other domains and to optimize its performance and usability for specific use cases. This paper presents an implementation of two image processing algorithms. Adaptive mean thresholding is a good option to improve the visibility of text for scanned documents. The median filter didn't show the desired results and made the text not readable. While it worked, for colored images or images of landscapes or portraits, it didn't for documents or images showing text.
The focus in the implementation was the enhancement of images of documents.

REFERENCES

[1] Zhenyun Zhuang, Sergiy Zhuk, Haricharan Ramachandra, and Badri Sridharan. 2016. Designing SSD-Friendly Applications for Better Application Performance and Higher IO Efficiency. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE. https://doi.org/10.1109/compsac.2016.94