HPES PDF Scanner
IEDP - Image Median Filtering & Edge Detection

Project Team:
Caide Lander – LNDCAI001 → Research and machine learning specialist
Lloyd Ross – RSSLLO001 → FPGA and Protyping specialist
Francis Mutetwa – MTTFRA005 → Golden Measure and algorithm specialist
Dante Webber – WBBDAN003 → Team Leader and algorithm specialist

April 20, 2024

Abstract

An independent embedded system that can take an image of a desired page, and after implementing the appropriate pre-processing such as noise reduction through median filtering and edge detection of the page and text for accurate, can convert the image to a PDF file and output it to a connected computer.

Figure 1: A Microsoft copilot interpretation of an FPGA PDF Scanner

Project Description

The aim of this project is to create an independent document-image processor using median filtering and edge detection.

At base level the device will be able to take in an image input - formatted in arrays of the pixel colour values by `width x height` (.bmp). The device will then process the image through appropriate algorithms including median filtering and edge detection, and finally convert the image to a PDF file.

Median Filtering

Median filtering is an image processing algorithm that calculates and re-populates pixel values in an image based on the median value of the surrounding pixels. This process helps to remove noise from images while preserving the edges of objects in the images. This is important for 'pdf' scanning because each printed item (word, letter, etc.) on a page is an object that will need to be detected later, so edges need to be preserved, and noisy images will make it harder to detect the valuable information (the objects). A flow chart detailing the concept behind the algorithm is shown below.

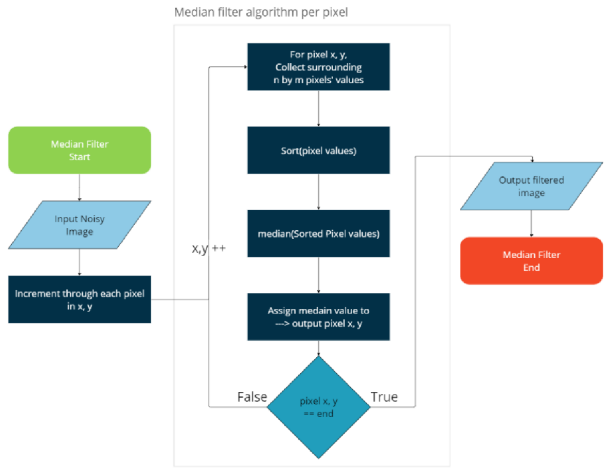

Figure 2: Median Filter Flow Chart

Edge Detection and Processing

Edges in images represent areas with significant changes in intensity or colours, marking transitions between different objects or regions within the image. Detecting these edges is crucial for computer vision and various image processing tasks, including document-image processing for accurate PDF generation [1].

When aiming to accelerate edge detection on an FPGA, it is necessary to prioritize algorithms that align well with the FPGA's hardware architecture. Given this requirement - the Sobel and Prewitt operators are favourable choices due to their simplicity and efficient parallelization potential on FPGA hardware [2]. Additionally, the Roberts Cross operator, with its straightforward computation, is suitable for FPGA implementation, particularly in resource-constrained environments.

Sobel and Prewitt Operators: These operators perform convolution with simple 3x3 kernels to compute the gradient magnitude and direction. Its straightforward computation makes it well-suited for FPGA implementation, where parallelism can be utilised to compute gradients for multiple pixels simultaneously [2].

Canny Edge Detector: While the Canny edge detector offers high-quality edge detection, its multi-stage algorithm and non-linear operations may pose challenges for FPGA implementation. However, with careful optimizations and efficient resource utilization, including optimizations for the non-maximum suppression

and hysteresis thresholding stages, the Canny detector can still be effectively implemented on an FPGA [3].

In addition to traditional edge-detection algorithms, the project aims to explore the potential of machine learning for edge detection. Machine learning has been used in various image processing tasks, including edge detection, by learning from large datasets to identify complex patterns and features within images. Integrating this into the image scanner and processor can enhance the system's ability to detect edges accurately, especially in scenarios with varying edge complexities and noise levels.

Connected Components

A connected component is an image processing term that refers to neighbouring pixels that all have an intensity value (for grayscale images) or colour (for coloured images) that falls within/above a custom defined threshold. This algorithm/tool can be used to find groups of connected pixels on a page, such as a letter (should consist of darker pixels than the page-colour pixels surrounding it). This will aid in text recognition algorithms, as the text can be isolated and then compared.

Prototype Specification

- The system will be initially simulated in Verilog to confirm the concepts functionality.
- The prototyping will then be done on an FPGA using the Nexys4 [4].
- Image to be processed must be converted into or stored in bitmap format (.bmp).
- There will be support for USB input and output communication.
- The device will be able to idle while waiting for a trigger (input).
- The device will be triggered when there is an image input sent to it.
- The device will be able to convert the image to a PDF file type.
- The device will be able to send the processed PDF file through the USB communication infrastructure.

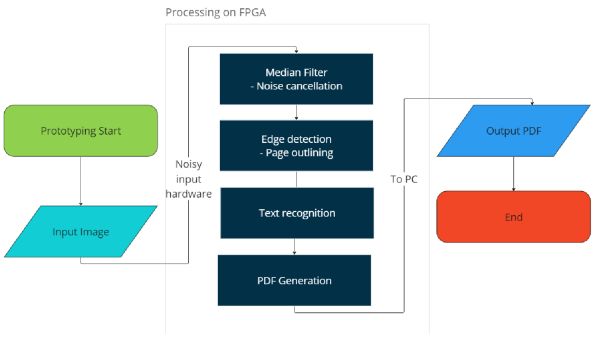

Figure 3: Base level prototype concept flow chart

Project Goals

- Starts up.
- Can interpret a bitmap image input.
- Can detect noise.

- Can filter irregularities due to noise using median filter.
- Can determine edges of page from image.
- Can recognize text.
- Can generate a pdf of page image.
- Can successfully output PDF to computer.
- Accelerate processes using parallelization.

If physical prototyping fails, the project goals are to be achieved on a thorough testbench simulation using HDL.

References

[1] D. Nagalakshmi and S. Jyothi, "Image acquisition, noise removal, edge detection methods in image processing using matlab for prawn species identification," *G Nagalakshmi, S Jyothi*, 05 2015.

[2] G. Chaple and R. D. Daruwala, "Design of sobel operator based image edge detection algorithm on fpga," in *2014 International Conference on Communication and Signal Processing*, 2014, pp. 788–792.

[3] Q. Xu, S. Varadarajan, C. Chakrabarti, and L. J. Karam, "A distributed canny edge detector: Algorithm and fpga implementation," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 2944–2960, 2014.

[4] D. Reference, "Nexys 4." [Online]. Available: https://digilent.com/reference/programmable-logic/nexys-4/starT