Edge-Based Segmentation on Maxeler

Christian Permann

Faculty of Computer Science, University of Vienna,
Währinger Straße 29, 1090 Vienna

Abstract This paper explains the basics of Edge-Based Segmentation and how it was trivialy implemented on a Maxeler architecture. This should result in a high speedup, since the calculations can benefit a lot from the DataFlow paradigm.

Contents

1 Introduction and Basic Idea

Edge-Based Segmentation is an image processing method that aims to find the outlines and shapes of objects within a picture. For this the image is usualy analysed in grayscale and a filter may be applied. Filters look at a window of size $n \times n$ (here 3×3) around the targe pixel and have different logic for choosing a pixel. There are three main filter variants fo be considered for this task.

High-Pixel Filter The High-Pixel Filter looks for the highes pixel value within the window and selects it for the target pixel.

Low-Pixel Filter The Low-Pixel Filter looks for the lowest pixel value within the window and selects it for the target pixel.

Median-Pixel Filter The Median-Pixel Filter looks for the median pixel value within the window and selects it for the target pixel.

After this filter is aplied the value of each pixel in the image is compared to a threshold, which is usualy chosen as the overall average or median pixel value, and either rounded to the white or to the black value bepending on if the value is higher or lower. Everything up to this point will from now on be to refered as the frist phase.

This results in a purely black and white representation of the image which suffices for easier edge and shape recognition for humans. To further analyse this for shapes it is now easy to compare each pixel with its neighbours and if not all of them are the same, the target pixel is part of an edge. With this logic a new image can be created form this one, which contains a certain color for edge pixels and white for non-edge pixels. This will be refered to as the second phase. The resulting edges may also still be improved with different methods.

2 Example

To further ilustrate how the edge segmentation works an image (Figure 1) will be used to show the different intermediate steps.

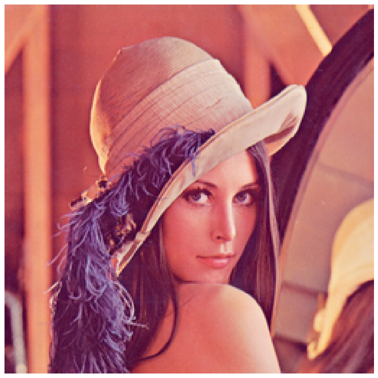

Figure 1: The example image used

First the example image is transformed into grayscale before any further transformation. (See Figure 2)

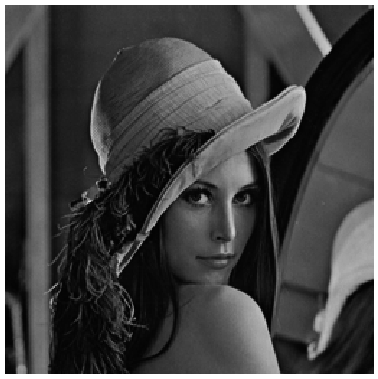

Figure 2: Grayscale transformation of the image

Next a low, high or median pixel filter (or no filter) is aplied and the result is rounded towards black or white, as can be seen in Figure 3. Here the grayvalue 120 was chosen for rounding, there may be a possible improvement in the result if the average or median between all pixels is chosen.

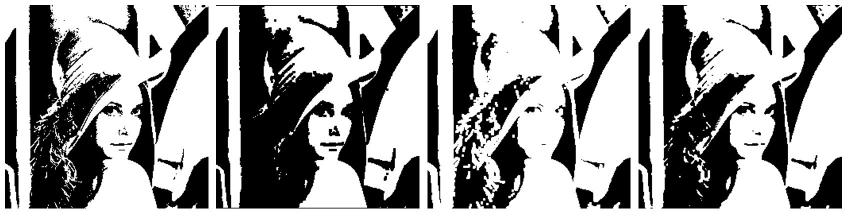
(a) No Filter (b) Low Pixel (c) High Pixel (d) Median Pixel

Figure 3: Different filters after rounding (120 as threshold)

The results of the edge calculations can be seen in Figure 4. This is the result of coloring every pixel with at least one differing neighbour pixel black.

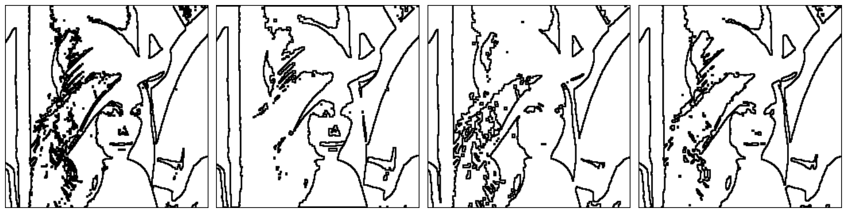
(a) No Filter (b) Low Pixel (c) High Pixel (d) Median Pixel

Figure 4: Different segmentations based on filters

3 Implementation

For this university assignment, the task was to extract one loop from the algorithm and implement it with Maxeller Java. For this, the filtering and rounding was chosen especialy because the median filter may increase the amount of work per iteration such that using the accelerator has a higher payoff.

The main challenges of implementing this were firstly dealing with the boundary cases when looking at the 3 by 3 window arround pixels on the border of the image. For this a counter was used and each access of a window element is checked if it still is within the bound, otherwise the center pixel is used because we know for sure that it is in the image bounds.

the second challenge was the median filter, the logic which finds the median within the pixel window, increases the complexity per array entry and may therefor improve the performance gain relative to running the code on the cpu.

It is also possible to accelerate the reduction to just edges, although it may not pay off as much because there is less work per loop iteration and the comunication with the accelerator may be more expensive then the performance gain.

The idea for the algorithm and code were taken from [1].

3.1 Maxeler Execution Graph

An execution graph is generated by the Maxeler compiler which visualizes how the FPGA will be reprogrammed. The execution graph for this example can be seen in Figure 5.

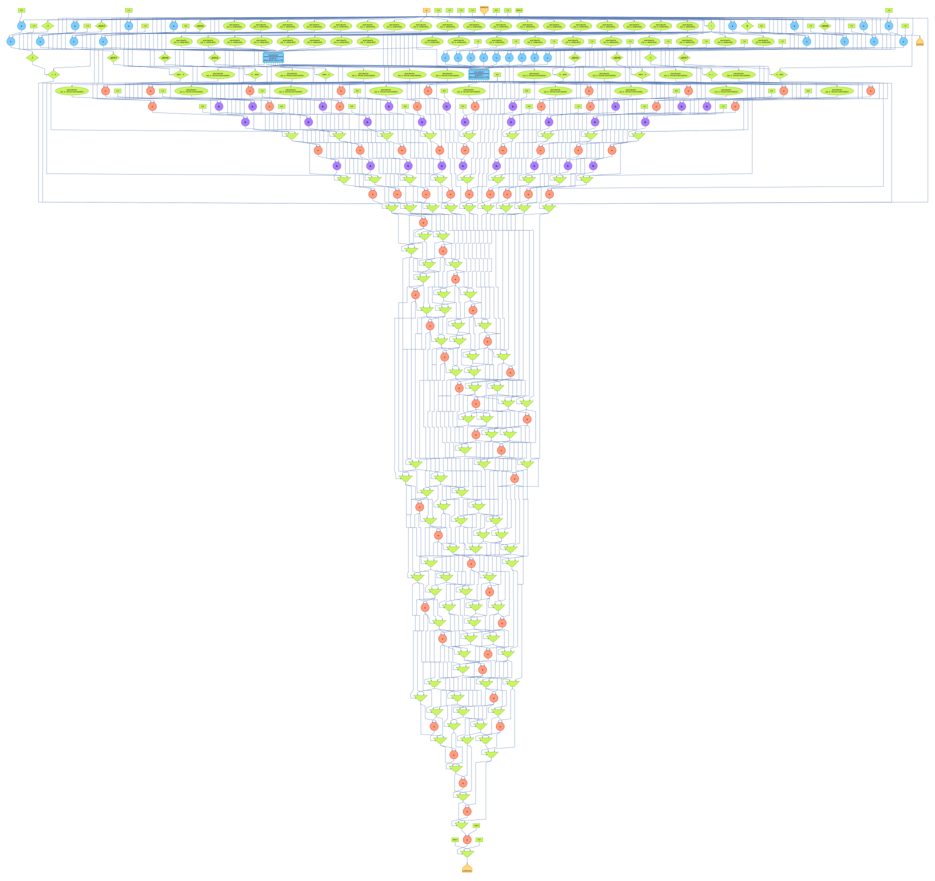

Figure 5: Execution graph for of the first phase of Edge Segmentation

4 Conclusion

The different filter types result in very different results after the edge segmentation. It is difficult to know for the general case which filter (if any) should be used and what exact threshold to pick for rounding, although the mean or median may be a good idea. The implementation on maxeler should improve the runtime performance, at least as long as the work within iterations is more relevant than the communication overhead from sending over data.

Acknowledgments

The author would like to thank Milos Kotlar and Prof. Veljko M. Milutinovic for their teachings on the subject of Maxeler and DataFlow computing.

References

1. Phillips, D.: Image processing, part 10: Segmentation using edges and gray shades. C Users J. 11(6), 67–98 (Jun 1993), `http://dl.acm.org/citation.cfm?id=157413.157422`