Fast and comprehensive FPGA based BLOB analysis with the Hybrid-BLOB concept

Simon Wezstein[1,2], Michael Stelzl[1], and Michael Heizmann[2]

[1] MSTVision GmbH,
Im Weiherfeld 10, 65462 Ginsheim-Gustavsburg, Germany
[2] Karlsruhe Institute of Technology,
Institute of Industrial Information Technology,
Hertzstraße 16, 76187 Karlsruhe, Germany

Abstract In this contribution we show our approach for a feature rich and high speed BLOB analysis on FPGAs. For the Hybrid-BLOB concept we use a combination of a single-pass BLOB analysis and a double-pass labeling algorithm. We use Basler's VisualApplets for the implementation of the concept on their microEnable 5 frame grabbers. We achieve the extraction of the gray value data of the BLOBs at factor 14 higher frame rates compared to the naive labeling of the complete image. This is achieved by limiting the maximum BLOB size to 128×128 px, which speeds up the double-pass labeling algorithm. Our concept is targeted at low latency and high throughput demanding applications where BLOBs are small, like sensor based sorting or surface inspection.

Keywords Image signal processing, FPGA, BLOB analysis

1 Introduction

In image processing the term Binary Large Object (BLOB) analysis refers to the extraction of connected components of a binary image with posterior calculation of the component's features like area, circumference, etc. The features are often used to classify these components. They are often called objects, as in many applications single objects are

segmented and analyzed. In inspection tasks these algorithms may be used for the classification of single objects, e.g. into “accept” or “reject” classes or to divide the defects even further, for example into “dent”, “scratch”, etc.

In image processing Field Programmable Gate Arrays (FPGAs) are used if high throughput, low latency or energy efficiency is demanded. For example FPGAs are used directly in cameras for post processing of the sensor data. They are also used in special applications like sensor based sorting or surface inspection.

MSTVision developed an FPGA based sensor based sorting platform, which aims at minimum latencies [1]. Its logic is completely implemented with VisualApplets (VA). VA is a proprietary development platform by Basler (formerly Silicon Software) for FPGA image processing logic development, tailored for their frame grabbers and devices with embedded VA support [2]. The platform proved its low latencies of around 200 μs in [3]. Currently the system’s image processing capabilities are limited by the feature limits of the VA BLOB analysis operator.

To run the mentioned tasks on FPGAs, implementations of the BLOB analysis are required. The research field in FPGA based BLOB analysis algorithms is still active. To extract BLOB features, first the connected components need to be extracted. This process is named labeling, its output is an intermediate image, with unique pixel values for each connected component in the image. There are many algorithms, but the algorithms may be divided in four categories [4, p. 352-359]:

1. Single-pass algorithms, where the data only needs to pass the computing pipeline once.
2. Double-pass algorithms, where the data needs to pass the computing pipeline twice.
3. Multi-pass algorithms, where the data needs to pass the computing pipeline multiple times, depending on the image content.
4. Random-access algorithms, where the data needs to be accessed randomly.

Each algorithm category poses its own pros and cons. Most of the current research focuses on single-pass algorithms, as they provide the

lowest possible latencies and demand only small FPGA resource quantities. The BLOB labeling is done only implicit. The downside is the limited amount of extractable features, which we will explain in the next paragraph. We will focus on single- and double-pass algorithms, as they are used in this contribution.

1.1 Labeling problem in detail

The main problem for image stream labeling algorithms are "U" shaped components, for example see fig. 1. While processing the binary image stream, the first object pixel is observed at (1,4). A new label is created for a unique representation of the object. In the next image line at (2,1) another object pixel is observed, but based on the processed data, it's not connected with the ones in the line before. A new label is created. While scanning the line, both labels coexist. In line 3 both labels turn out to be connected at (3,3) or (3,4), depending whether the 4-connected or 8-connected neighborhood is used. This results in a problem: the previously assigned labels need to be merged into one. The way the algorithms overcome that problem is their fundamental difference.

Double-pass algorithms like [5, p. 4] use equivalence tables to record these conflicts. One object may consist of many intermediate labels. After the first labeling pass, a conflict resolving algorithm is used to convert the labels to a unique final label lookup table (LUT). With the LUT and the result image of the first pass, the final label image is created. The advantage over single-pass algorithms is the ability to extract the component pixel accurately. This enables the calculation of all object features after labeling. The disadvantages are their higher memory demands for buffering the intermediate label image and the equivalence table. Resolving the label conflicts and calculation of the features after the labeling adds computing time. For FPGA implementations the often required random memory accessibility for the equivalence table is a limitation, too.

Single-pass algorithms like [6] don't provide a label image output, instead they calculate the object features directly. The labeling is only carried out internally. A single-pass algorithm performing the extraction of the object area would work on the example in fig. 1 as follows: the first object pixel is observed at (1,4), a new temporary label and ac-

cumulator is created. For each connected pixel the area is incremented by 1. In the next image line at (2,1) another object pixel is observed, another temporary label and accumulator is created. When both labels collide, one label is deleted and its area accumulator is added to the other accumulator. The output of the algorithm is a list of component features, in this example only the area. There is no ability to extract the object pixels to compute other features. The features which may be extracted are limited to those which may be merged out of the values of sub component features on label collision. Their advantages are the small memory requirements, which is limited to the feature and label table, and the smaller computing time.

With single-pass algorithms features like the oriented bounding box or minimum/maximum Feret diameters can't be calculated. These features are usually calculated with the object's convex hull and the rotating calipers algorithm. [7]

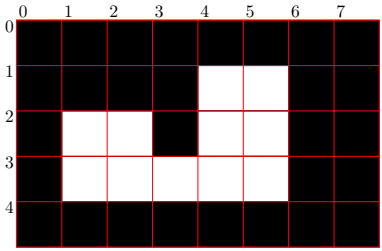

Figure 1: A simple "U" shaped object to demonstrate the main challenge of streaming labeling algorithms. Modified version from [8, Fig. 3]

2 Method

To fill the gap between single- and double-pass algorithms, we developed the Hybrid-BLOB concept. The method consists of two BLOB analysis/labeling algorithms, a single-pass algorithm and a double-pass algorithm. The single-pass algorithm is the one used in the VA BLOB analysis operators [9]. The double-pass algorithm is our implementation of the algorithm described in [8].

The double-pass algorithm is expensive with respect to computing

time and memory if applied to a big image. For big images, the conflict resolve table won't fit into the FPGA's on-chip memory of current Basler frame grabbers, requiring utilizing the off chip DRAM. Resolving the conflicts is an algorithm of quadratic order. The single-pass algorithm in comparison does only provide a few features.

To overcome the limitations, both algorithms are combined, as described in the next subsection. This allows smaller input image sizes for the double-pass algorithm, thus the conflict resolve table fits into the FPGA's on-chip memory and the conflict resolve algorithm may run faster.

2.1 Architecture overview

In fig. 2 the concept is shown. The image input is preprocessed and segmented. The single-pass BLOB analysis of VA is applied to the segmented image. In parallel, the segmented image and the preprocessed gray image are stored into dynamic random access memory (DRAM). A pre-classification is applied to the output of the single-pass BLOB analysis. The remaining objects of interest are retrieved from the DRAM buffer via their bounding box information. The extracted object images may contain pixels of other objects, as shown in the example BLOBs in fig. 2. The double-pass algorithm is then used to label the small images. With the bounding box information of the previous BLOB analysis and the label image, the corresponding object may be extracted from the binary and gray image. Afterwards we extract various object features which then may be used for object classification.

2.2 Implementation

The implementation is done in VA with only VA operators except one VHDL custom operator. The target hardware platform are the microEnable 5 marathon frame grabbers, [11]. As the implementation of most of the single architecture elements is straightforward, we focus on the double-pass labeling algorithm and the feature extraction. For comparison we use an implementation of the labeling algorithm for the labeling of a whole $1024 \times 1024\,\mathrm{px}$ image. The maximum configurable bounding box size for labeling in Hybrid-BLOB is limited to

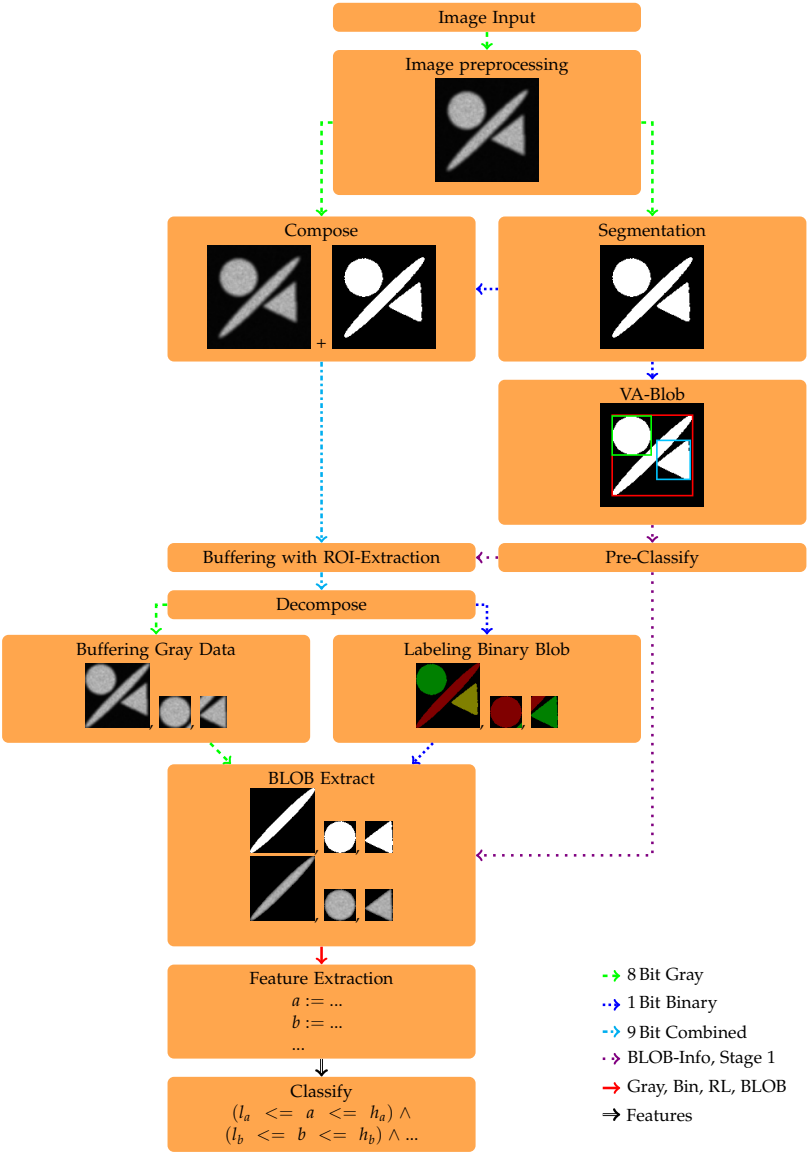

Figure 2: Hybrid-BLOB architecture overview. Modified version from [10, Fig. 4.1]

Table 1: Comparison of memory requirements of the implementations. The maximum code length was empirical determined. *id* is the label id, *eq* is the equivalent label id in case of a conflict, *r* is the run element's row, *s* and *e* are start and end column of the run. *BRAM* is for Block Random Access Memory, the FPGA's on-chip memory. [10, Tab. 4.1]

Parameter	Labeling	Reduced Labeling
id, eq	16 Bit	8 Bit
r, s, e	13 Bit	7 Bit
Memory per element	10 Byte	5 Byte
Max. label count	65535	255
Max. code length	65535	4095
Maximum memory	5.24 MBit	163.8 kBit
BRAM-Elements (18 kiB per element)	291.3	9.1

128×128 px. The labeling stage uses fixed frame size inputs of the configured maximum bounding box size. The design transfers the input image and an image with the BLOB features over Direct Memory Acess (DMA) channels.

Labeling The labeling algorithm is an implementation of [8]. The algorithm is a run length encoding (RLE) based, 4 connected neighbourhood type. Depending on the design, other bit depths are used for the labels and the run length code. Labeling smaller images results in smaller coordinate bits and fewer possible labels. In tab. 1 the resource occupation for both variants are shown. By reducing the image size which has to be labeled, the required memory drops, which practically enables the storage of the run length data in the FPGA's on-chip memory. For the labeling of the whole image, the data is stored in the frame grabber's DRAM. The whole image labeling design does not contain the calculation of features.

Feature extraction With the extracted object's image data, the feature extraction takes place. The extraction is completely integrated into the FPGA. The orientated bounding box and Feret features are not calculated with the convex hull and rotating calipers. They are approximated by discrete object rotations in angle steps of $0.703\,^{\circ}$. To save FPGA resources, the calculation of unneeded features may be removed.

The extracted features are:

- VA-Operator: *bounding box, area, center of gravity* (output of single-pass analysis stage).
- Gray Value: *min, max, mean, std, median, upper and lower quartile, difference to a reference histogram (rel/abs)*.
- Other binary image features: *Euler's number, circumference, compactness, circularity, circle equivalent diameter*.
- Binary image moments: *2nd and 3rd order*.
- Ellipse features: *main axis angle, main and minor axis radius, eccentricity*.
- Gray image moments: *2nd and 3rd order*.
- Oriented bounding box: *area, angle, width, height*.
- Feret diameter: *minumum, maximum, min. angle, max. angle*.

For further information about the features, we suggest [12], [13], [14], [15] and [16].

3 Results

Both test designs have been built with VisualApplets 3.3.1 for the microEnable 5 Marathon VCLx frame grabber, running at a frequency of 125 MHz [17] [18]. The used frame grabber runtime is version 5.7. In fig. 3 our test image is shown. It contains 1161 BLOBs and has a resolution of 1024×1024 px. The amount of objects is not representative for real applications. The image is uploaded to the FPGA DRAM and repeatedly processed for our tests. We use the shown frame rate of microDisplay, the runtime application used to configure the frame grabber. The frame rates are validated against debug registers on the FPGA. The BLOB frequency is measured with a debug register. For our measurements, we don't filter the output of the single-pass stage.

The BLOB count varied from 1143 to 1161 while testing. The reason of these variations is currently unknown, their origin is the single-pass

Table 2: Measurement results for processing the image shown in fig. 3.

Parameter	Full Labeling	Hybrid-BLOB
Frame rate	1.25 Hz	17.0 ± 0.5 Hz
Mean BLOB frequency	1.5 kHz	19.8 ± 0.010 kHz
Mean time per BLOB	689 μs	51 μs
Labeling throughput	1.3 Mpx/s	324.4 Mpx/s

stage. We use 1161 as BLOB count for calculating the mean time of the labeling design and the period of the BLOB frequency for the Hybrid-BLOB design. In tab. 2 our results are shown. Our Hybrid-BLOB concept runs at 14 times higher frame rates even with the 250 times higher data throughput in the double-pass stage. Due to the fixed frame size for bounding box extraction, the labeling overhead increases if many small objects are present.

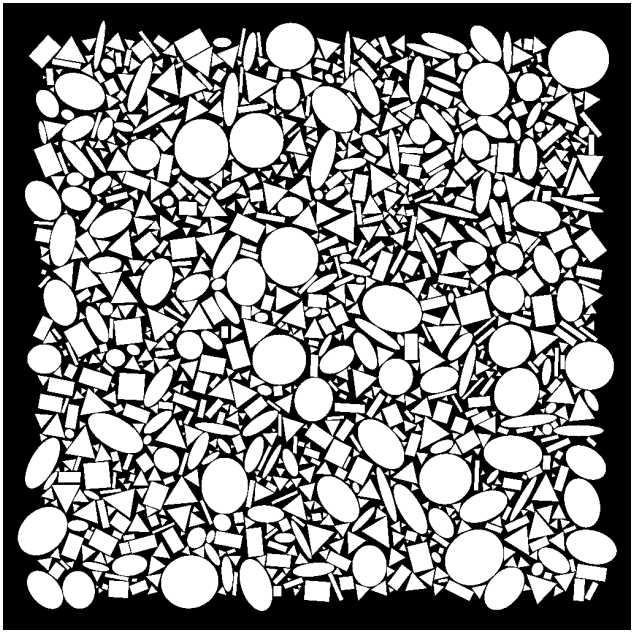

Figure 3: Test image used. It contains 1161 objects at a resolution of 1024×1024 px.

4 Conclusion

We have shown an approach to speed up a feature rich BLOB analysis on FPGAs. The implementation with VisualApplets enables the usage on Basler frame grabbers of the current portfolio and possible future platforms supporting VisualApplets. Hybrid-BLOB processes in our test scenario 19.800 BLOBs per second, which allows its usage in the field of granule sorting. To increase the throughput further, the labeling and feature extraction stage may be implemented multiple times in parallel. Our concept may be used in traditional PC based image processing, too.

The throughput and latency may be further improved if the double-pass labeling algorithm is extended to support variable image input sizes for overhead reduction. If variable input sizes are used, the run length encoding stage runs faster and the count of runs to label decreases. We expect big improvements if small BLOBs are processed, as the measured overhead is 250 times compared to the image input.

Acknowledgments

The work described here results from the project “Hybride Bildverarbeitung auf FPGAs”, supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag, for which we are grateful. We want also to thank Basler AG for giving the opportunity for the work described in [10], which is the origin of our concept.

References

1. MSTVision GmbH, “Minimale Responsezeit,” https://mstvision.de/downloads-sorting/, 2020, online, accessed 25-September-2022.
2. Basler AG, “VisualApplets Graphic FPGA Programming,” https://baslerweb.com/en/products/visualapplets/, 2022, online, accessed 15-September-2022.
3. S. Wezstein, M. Stelzl, and M. Heizmann, “Latency evaluation of an FPGA-based sorting system,” in *9th Sensor-Based Sorting & Control 2022*, Greiff, Kathrin and Wotruba, Hermann and Feil, Alexander and Kroell, Nils and

Chen, Xiaozheng and Gürsel, Devrim and Merz, Vincent, Ed., 04 2022, pp. 143–160.

4. D. G. Bailey, *Design for Embedded Image Processing on FPGAs*. Singapore: John Wiley & Sons (Asia) Pte Ltd, 2011.
5. Y. T. Kong and A. Rosenfeld, *Topological Algorithms for Digital Image Processing*. Amsterdam, The Netherlands: Elsevier Science B.V., 1996.
6. D. Bailey and C. Johnston, "Single pass connected components analysis," *Proceedings of Image and Vision Computing*, 01 2007.
7. G. Toussaint, "Solving geometric problems with the rotating calipers," in *Proceedings of MELECON '83, Mediterranean Electrotechnical Conference*, 1983.
8. K. Appiah, A. Hunter, P. Dickinson, and J. Owens, "A run-length based connected component algorithm for FPGA implementation," in *2008 International Conference on Field-Programmable Technology*, Dec 2008, pp. 177–184.
9. Basler AG, "Library Blob," https://docs.baslerweb.com/visualapplets/files/manuals/content/library.Blob.html, 2022, online, accessed 16-September-2022.
10. S. Wezstein, "FPGA basierte Blobanalyse mit dem „Hybrid-BLOB"-Verfahren," Master's Thesis, Hochschule Darmstadt, 2021.
11. Basler AG, "microEnable 5 marathon Frame Grabber Portfolio," https://www.baslerweb.com/en/products/acquisition-cards/microenable-5-marathon/, 2022, online, accessed 15-September-2022.
12. B. Jähne, *Digitale Bildverarbeitung*, 7th ed. Berlin: Springer Vieweg, 2012.
13. M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. USA: Thomson-Engineering, 2007.
14. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, fourth edition, global edition ed., Array, Ed. New York, NY: Pearson Education Limited, 2018.
15. J. Ohser, *Angewandte Bildverarbeitung und Bildanalyse*. Leipzig: Carl Hanser Verlag GmbH & Company KG, 2018.
16. R. M. Haralick and L. G. Shapiro, *Computer And Robot Vision*. USA: Addison-Wesley Publishing Company, Inc., 1992.
17. Basler AG, "VisualApplets 3 User Manual," https://docs.baslerweb.com/visualapplets/files/manuals/content/device%20resources.html, 2022, online, accessed 15-September-2022.
18. ——, "microEnable 5 marathon VCLx - Frame grabber," https://www.baslerweb.com/en/products/acquisition-cards/microenable-5-marathon/microenable-5-marathon-vclx/, 2022, online, accessed 15-September-2022.