BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

HARDWARE ACCELERATION OF OBJECT DETECTION IN IMAGES

HARDWAROVÁ AKCELERACE DETEKCE OBJEKTŮ V OBRAZE

PHD THESIS

DISERTAČNÍ PRÁCE

AUTHOR **Ing. PETR MUSIL**

AUTOR PRÁCE

SUPERVISOR **Prof. Dr. Ing. PAVEL ZEMČÍK**

ŠKOLITEL

BRNO 2020

Abstract

This dissertation aims to propose methods for optimising the object detector in an image running on an FPGA. These detectors use boosted soft cascades of classifiers with local image feature like weak classifiers. The proposed detectors use sequential evaluation of weak classifiers. More positions in the image are evaluated in parallel to increase the detection performance. Also, a new approach for multiscale object detection is proposed; its advantage is no need for external memory. The new detectors were experimentally verified on the tasks of detecting faces and license plates. The results outperform the current state-of-the-art, allow to create object detectors with higher detection performance, better power to resources ratio and better detection accuracy.

Abstrakt

Cílem této disertační práce je představit navržené metody optimalizace detektoru objektů v obraze běžících na FPGA. Tyto detektory využívají boostovatelné soft kaskády klasifikátorů spolu s lokálními obrazovými příznaky, které slouží jako slabé klasifikátory. Navržené postupy využívají sekvenční vyhodnocení slabých klasifikátoru. Pro zvýšení výkonu detekce je vyhodnocováno současně více pozic v obraze. Je navržen nový přístup pro detekci objektů různé velikosti nevyžadující externí paměť. Vytvořené detektory byly experimentálně ověřeny na úlohách detekce obličejů a poznávacích značek automobilů. Dosažená výsledky překonávají současný stav poznání, umožňují vytvořit detektory objektů s vyšším detekčním výkonem, lepším poměrem výkonu a spotřebovaných zdrojů FPGA a s lepší přesností detekce.

Keywords

Object Detection, AdaBoost, WaldBoost, Acceleration, FPGA

Klíčová slova

Detekce objektů, AdaBoost, WaldBoost, Akcelerace, FPGA

Reference

MUSIL, Petr. *Hardware acceleration of object detection in images.* Brno, 2020. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Prof. Dr. Ing. Pavel Zemčík

Hardware acceleration of object detection in images

Declaration

Hereby I declare that this dissertation thesis is my original work and that I have written it under lead of Prof. Dr. Ing. Pavel Zemčík.All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Petr Musil
August 30, 2020

Acknowledgements

I would like to acknowledge and thank the following important people who have supported me, not only during the doctoral study. First of all, to Pavel Zemčík, my supervisor, for his guidance, knowledge and unwavering supports. To my friends for their personal and professional support, especially to Michal Hradiš, Roman Juránek and Martin Musil. I would also like to thank my family and my girlfriend, Karolina, for her support, language corrections and love.

Contents

Chapter 1

Introduction

Nowadays, we encounter an increasing number of cameras and surveillance systems. We can see cameras at toll gates, security cameras in buildings or police surveillance systems. The amount of information that these devices produce is enormous, and it is not in human power to process and interpret it all. The only option is to use computing power to analyse the huge number of videos and frame sequences. Modern computer vision algorithms have passed the point, where it is reasonable to start implementing them widely. Algorithms for object detection and recognition, for example of human faces, pedestrians, cars, or traffic signs already outperform human. In general, one of the disadvantages of advanced vision algorithms is high computational complexity. For this reason, it is necessary to use powerful computer systems with high energy consumption and cost. Also, it would be convenient to process most data locally without the need for remote servers, so-called edge computing.

One solution can be the hardware acceleration of computer vision algorithms on FPGA or ASIC chips. The aim is to create low-cost, low-power devices for real-time video processing. The deployment of algorithms to FPGA and ASIC circuits is specific and differs greatly from deployment to conventional computing

systems. Usually, a direct implementation of computer vision algorithms without their modification is inefficient, slow and resource-intensive. The acceleration of computer vision algorithms in hardware has long been the goal of many scientific works. This topic is attractive due to its potential practical application, and a combination of different research fields: image processing and hardware acceleration.

This thesis presents my contributions to the state-of-the-art in the topic of visual object detection in FPGA. Specifically, the work is focused on fast and powerful object detectors with low demands on resources. Such detectors could be applied mainly in transport, industry or security.

My contribution is in proposing methods for optimizing object detection on FPGAs. The main focus is on detectors using boosted soft cascades of classifiers with local image features as weak classifiers. Sequential evaluation of weak classifiers has been upgraded with parallelization by evaluation of several independent image positions simultaneously. Also, a new approach for multi-scale object detection has been proposed; its advantage is no need for external memory. Using these methods to create effective detector verifies the hypothesis: that it is possible to design an object detector based on soft cascade deployed in programmable hardware with resulting precision comparable to the state-of-the-art, with real-time performance, with lower power consumption and less computing resource demands comparing to existing ones.

Chapter 2

Goals and Contributions

This thesis focuses on object detection in images on hardware platforms. The scope of the work is to shift the scientific knowledge and apply it into the practice.

Presumed usage of object detectors developed here is industrial, transport or security applications, i.e. in tasks such as the detection of faces, pedestrians, products, licence plates. Practical deployment of such object detectors needs to meet specific requirements. The resolution of the processed image is an important parameter. This resolution depends a) on the resolution of the input image, b) on the expected size ranges of the searched objects and c) on the size of the detection window. Thus, the expected object size ranges significantly affect the required performance of the detector. Traffic and security applications require approximately 10 to 20 frames per second for suitable object tracking. Industrial applications often require even higher processing speeds. High-speed detectors also allow for processing image data in the camera without storing them on a fast external memory; the absence of external memory further reduces the price of the resulting device.

A typical requirement is that the detectors should achieve the best accuracy possible. In general, object detectors often balance

a speed-precision trade-off. Detection of visually diverse, rotated, or distorted objects will either be less accurate or will require a complex system with a large number of computational resources. For example, in the task of licence plates detection in toll gates, where the approximate size and rotation is known, excellent accuracy can be expected. However, a similar task, licence plates detection for parking control in residential zones, is more challenging due to unconstrained conditions (variable position, scale, rotation, etc.), and therefore lower accuracy can be expected.

The technical goal is to create a powerful universal object detector for FPGA hardware with good accuracy and low resource consumption. Such detector should process at least FullHD video at 15 frames per second and should detect small and large objects with limited variability. The detector should achieve accuracy (recall with precision 0.70) of at least 95% for face detection and 99% for LP detection. Important parameters for the low price of the device are low power consumption and a small number of FPGA resources used. The detectors are expected to be utilised in smart cameras. Such smart camera should perform the maximum number of operations directly in-site and send out only the results for further processing. Thanks to this, it would be possible to reduce the data flow from the camera as expected by edge computing.

2.1 Technical implications

At the beginning of my research work, there were already several successful attempts to create an object detector in hardware [1, 2, 3, 4, 5, 6, 7, 8]. A detailed overview of the individual solutions was given in the previous chapter. In summary, the results show that the practical use of these detectors is limited. In order to increase the performance and accuracy of detectors and the resolution of the processed image, it is necessary to improve current detection methods in hardware, modify detection algorithms and apply hardware-specific features.

Choice of classifier algorithm

Hardware detectors based on boosted classifiers have the most advantages for the applications mentioned above. The specifically targeted ad-hoc detectors have an excellent power-to-resource ratio in some applications. However, the ad-hoc detectors are not universal, the accuracy of detection is low, and they do not cope with changing conditions. Creating a detector for a new object class means a lot of work and an uncertain result. The SVM based detectors have good performance and relatively low resource consumption. However, they provide low detection accuracy not applicable in modern applications.

CNN based detectors provide the best accuracy and excel at complex detection task. Due to a large number of operations, they have high demands on logic and memory resources and have relatively low performance. Besides, the CNN optimisations for FPGAs means a loss of accuracy down to the level of a modern boosted classifier. Considered tasks such as face/pedestrian detection and licence plate detection have only a small number of object classes with a relatively consistent appearance, for which CNN seems unnecessarily complex. In conclusion, the modern boosted classifiers provide sufficient accuracy and speed trade-off considering these tasks with high-throughput and resource-limited scenarios.

Individual modifications of boosted classifiers, such as soft-cascade or Waldboost [9, 10, 11], only differ in the training process. The evaluation step does not vary much; the only difference is that soft-cascades and Waldboost allow rejecting after each weak classifier and the original Viola-Jones algorithm [9] allows rejecting after each stage (a set of several weak classifiers). The planned detector should evaluate all modifications of boosted classifiers. The best way to reach high detection accuracy and performance is training the boosted classifiers with Waldboost algorithm [11] and with augmentation, as suggested by Bar et al. [12].

Two approaches for implementation of boosted classifier based detector were developed — fully pipelined monolithic detector [7, 13] and sequential detector [8, 6, 4]. Fully pipelined detectors assess all features for one detection window position in parallel per one clock cycle. In general, the fully pipelined detectors are high-throughput and easy to implement. However, they have high demands on FPGA resources and usually evaluate only a limited number of weak classifiers, which leads to lower accuracy. Sequential detectors assess the features gradually. The parallelisation of sequential detectors is possible by processing multiple features or multiple positions at the same moment. They require fewer resources and allow for better accuracy, but they are more challenging to control. I have focused on sequential detectors because they meet the required parameters better and offer space for further development.

Choice of weak classifiers

Many authors [2, 3, 1, 4, 5, 6] use Haar-like features as weak classifiers. High bit depth for storing integral image and complex logic to access feature values make them resource demanding. Therefore, it isn't easy to increase the performance of such detectors. Other works [8, 7, 13] apply LBP or their modifications (LRD, LRP) as weak classifiers. Their advantage is in loading only surrounding pixels (block reading) for effective calculation. LBP-like features are mainly used in combination with fully pipelined detectors [7, 13]. Zemčík and Žádník [8] verified the sequential approach by developing a suitable object detector. Their detector used LRD features with a size of 3x3 and subsampling of the original image in the ratios 1x1, 1x2, 2x1 and 2x2. For direct evaluation of the feature, it may require loading blocks of up to 6x6 pixels. It seems unnecessary to read data from the sliding windows (usually created as a register array with FIFO line buffers), and a better option is to read it directly from the addressable memory composed of BRAM. An optimal structure of this BRAM memory allows data reading

without the use of a complicated multiplexer network. Zemčík and Žádník reduced the logic resources required for reading such large blocks by precalculating the subsampled versions of the image into memory. Reading only 3x3 blocks already precalculated in memory becomes sufficient for feature evaluation. The disadvantages are increased memory requirements and the need for storing differently sized subsampled versions of the original image. This has led to a complicated memory structure and high logic and memory resources demands.

Hereby proposed detectors are inspired by the detector introduced by Zemcik and Zadnik [8]. The main difference is that local image features of different sizes are not calculated from precalculated values, but directly from the original image. It saves memory resources, but on the other hand, it means reading blocks of different sizes (3x3, 3x6, 6x3 and 6x6 pixels), which is more complicated compared to the original constant size of 3x3. For simplicity, a 6x6 block (the worst case) can always be read and then subsampled as needed. A well-designed memory structure allows reading of unaligned 6x6 blocks at the same time, thus reducing logic and memory requirements.

Multiscale-detection

Designing an effective multi-scale detection on FPGA is an unresolved issue. Many related works do not address multi-scale detection at all [14, 2, 8, 13]. Several works [1, 3] suggest storing the entire image in BRAM memory, but this is not always feasible, especially at higher resolutions. Other works solve multi-scale detection by multiple image loading from external memory [5, 6]. Kyrkou et al. [4] reduced the number of loading from external memory by using more classifiers with different window sizes.

We have proposed a more efficient method of multi-scale detection. It does not require multiple image readings and uses significantly less memory than needed when storing entire images. The core is that the single scale detection requires only a narrow strip

of the image memory, with the minimum height as the detection window. The same principle can be used for smaller versions of the image in multi-scale detection. Furthermore, it enables generating these smaller versions from the previous ones with fixed scale unit on-the-fly. The proposed method allows detecting objects of different sizes directly from the data coming from the sensor; the resulting system may not contain external memory at all, which would reduce the cost of the device. This approach can be further combined with the use of multiple classifiers with different window sizes. However, the memory requirements for storing multiple classifiers using LBP/LRD features are often greater than the memory consumption when storing stripes of image. But the benefit always depends on the resolution of the input image, the height of detection window and the number of down-scaled versions.

Detector optimization

Parallel processing is one way to increase the performance of the detector on FPGA. Many authors [1, 14, 4] use parallel computation of more features in one position. Since the average number of evaluated features in one position is very small, it does not allow a high level of parallelization. When using the sliding window approach, it is necessary to evaluate all weak features and only then it is possible to move to the next one. In pipeline processing, premature rejection often results in a penalty meaning a speculative evaluation of other features or insertion of blank operations. Brousseau and Rose [6] suggested a method of evaluating features in neighbouring positions. The number of evaluated classifiers in specific positions is variable, which causes problematic divergence of the calculation. Besides, this approach combined with sliding windows leads to an increase of multiplexing network complexity, and thus an increase in logical resources. The detector introduced by Zemčík and Žádník [8], which is the basis for the proposed detectors, did not use any parallel processing.

We have proposed an approach for evaluating multiple positions in the pipeline simultaneously. This is possible by reading the data directly from BRAM memory, where the data for evaluating all positions of the entire line are accessed. It enables us to evaluate a bigger number of independent positions at the same time at various stages of evaluation. After evaluating the required features in one position, there is no need to wait for the evaluation of the surrounding positions; it is possible to move to the next position in the same line. This eliminates the issue with divergence and allows the creation of a longer pipeline without penalizing after the early rejection. The extension of the pipeline has a positive effect on the increase of the maximum circuit frequency and consequently, the rise of the detector performance.

We also suggest using more detectors connected in a cascade to increase performance. Each of these detectors performs detection in a different part of the image and at a different scale version. The optimal distribution can be precalculated so that the number of positions evaluated by each detector is approximately the same. This modification allows the detector's performance to be scaled very well for the needs of a specific application.

2.2 Goal of the thesis

The primary goal of this thesis is to improve the state-of-the-art in the field of object detection in the image on hardware platforms. The hypothesis is: *It is possible to design an object detector based on soft cascade deployed in programmable hardware with resulting precision comparable to the state-of-the-art, with real-time performance, with lower power consumption and less computing resource demands comparing to existing ones.*

The method of proof is creating a hardware detector that meets the required parameters and thus exceeds the state-of-the-art. Completing this task will require developing new methods and

performing many experiments. In order to investigate of task the detector design, I have chosen to pursue the following methods:

- using local image features (LBP/LRD) and soft cascade classifiers in sequential engine with efficient block reading of image values for weak classifier evaluation,
- creating a multi-scale on-the-fly detector for high-resolution image data processing (without the need for external memory),
- using parallelization of weak classifier calculations by processing multiple positions at the same time, both at the level of sequential engine and cascade connection of multiple detectors.

The above proposed methods are being examined with the aim to confirm the presented hypothesis. The experiments will be performed in the detection of faces, pedestrians and license plates. Comparisons with other authors will be made on the face detection task, which is usually presented on other papers. For a fair comparison of performance due to different resolutions, detection window size, detection stride, multi-scaling, etc. a conversion to the number of processed detection windows per clock cycle will be used.

2.3 Core contributions

This thesis contributes to the state-of-the-art in the field of object detection in the image on hardware platforms. Three papers validating the hypothesis were published. They represent the experimental proof and demonstrate that it is possible to create the detector with defined parameters. The papers show that the proposed hardware detector outperformed the state-of-the-art in several aspects:

- better detection performance among boosted classifier – multi-scale face detection on Full HD (1920×1080 pixels) video at 60 fps (for object size 21 pixels and more) versus 640×480 at 30 fps by Hiromoto et al. [3],

- better detection performance in processed detection windows per clock cycle among all hardware detectors – up to 2.33 versus only processing detector with 1.97 by Jin [7] of full detector with 0.95 by Zemčík and Žádník [8],

- better performance/resources ratio – in all resources: LUT, REG and BRAM; the graphical comparison is in paper [15],

- better accuracy in face detection on CMU dataset [9] – recall 97 % with 0.2 false positives per image (FPPI) versus recall 91 % with 0.2 FPPI presented by Hiromoto et al. [3] and Kyrkou et al. [4],

- comparable accuracy in licence plate detection – aligned licence plates recall 99 % with 0.2 FPPI and unconstrained recall 98 % with 0.2 FPPI on own dataset

The contributions that implement experimental proof of thesis were presented in the following papers:

- *High performance FPGA object detector: Hardware prototype*, FPL[1] 2013. Paper with introduce an architecture of an engine for high-performance multi-scale detection of objects in videos based on WaldBoost training algorithm. The key properties of the architecture include the processing of streamed data and low resource consumption. The engine is implemented in FPGA and that it can process 640×480pixel video streams at over 160 fps without the need of external memory.

[1] International Conference on Field Programmable Logic and Applications

- *Cascaded Stripe Memory Engines for Multi-Scale Object Detection in FPGA*, TCSVT[2] 2019. Evolution of the previous paper witch expands performance and usability. FPGA detector can process a stream of image data so that it stores a narrow stripe of the input image and its scaled versions and uses a detector unit which is efficiently pipelined across multiple image positions within the memory. We show how to process images with up to 4K resolution at high framerates using cascades engine. As a detector algorithm use boosted soft cascade with simple image features that require only pixel comparisons and look-up tables; therefore, they are well suitable for hardware implementation.

- *Unconstrained License Plate Detection in FPGA*, submitted to VEHITS[3]. This paper shows the practical use of the previous detector in traffic application on the task of detecting unconstrained License Plate. To detect and localize license plates is use multiple sliding window detectors based on simple image features, each tuned to a certain range of projections. On a large dataset is detection rate 98%.

Results presented in these papers proof the hypothesis of this thesis.

2.4 Other publications

I am a co-author of some other publications dealing with other areas of image processing. I focused mainly on the effective implementation of the algorithm on the FPGA and the modification of the algorithm for stream image processing on the fly, ie without the use of external memory. List of my other publications in chronological order:

[2]IEEE Transactions on Circuits and Systems for Video Technology

[3]International Conference on Vehicle Technology and Intelligent Transport Systems

- *Single-Loop Approach to 2-D Wavelet Lifting with JPEG 2000 Compatibility*, SBAC-PADW[4] 2015 [16]. In this paper is presented a novel approach to 2-D single-loop wavelet lifting with can be efficiently pipelined in hardware. A newly developed 2-D core of CDF 5/3 wavelet filter is presented that, using a new sequence of operations, simplify the design. Moreover, the proposed approach, that uses one pass for 2-D transform, directly produces final output and reduces significantly the need for storing intermediate results into memory.

- *High Dynamic Range Video Concepts, Technologies and Applications*, Real-Time HDR Video Processing and Compression Using an FPGA, 2016 [17]. The chapter in the book deals with hardware acceleration of HDR video acquisition and compression. Individual HDR images are obtained by composing several differently exposed images obtained with a standard camera. Description of HDR compression and its implementation on FPGA.

- *True HDR camera with bilateral filter based tone mapping*, SCCG[5] 2017 [18]. In paper is presented a real-time HDR processing system evaluated on a custom hardware camera platform. They are proposal modifications of the the State-of-the-arts algorithms enabling efficient implementation on FPGA platform and real-time performance. The main focus of the paper is on acceleration of Durand local tone-mapping operator involving real-time bilateral filter. The proposed solution is compared to the existing research results in terms of speed, resource consumption, and numerical accuracy.

The publications presented above do not directly contribute to the scientific goal and hypothesis validation of the dissertation.

[4] International Symposium on Computer Architecture and High Performance Computing Workshop

[5] Spring Conference on Computer Graphics

However, technologically they add to the options of using object detection in images. In some applications, for example, in bad light conditions such as sharp backlight, it is advantageous to combine object detection with HDR image processing to improve accuracy. In addition, the platforms created in these publications were used for experimental work with object detection.

Chapter 3

Multi-Scale Object Detection in FPGA

Object detection in embedded systems is an important task that many applications of computer vision and scene analysis benefit from. Industrial quality control systems address various markers, traffic monitoring uses detection of cars and license plates, biometric systems detect faces and facial features, driver assistance systems detect cars and pedestrians. The detection is especially important in applications that directly rely on it, such as recognition or tracking, and in these applications, the speed, accuracy, power consumption, and/or robustness of detection matters most. In this work, we address object detection implemented in embedded hardware. We focus on boosted detectors which analyze sub-windows of an input image by a classifier composed from weak classifiers based on simple image features such as Haar [19] or Local Binary Patterns (LBP) [20]. Multi-scale detection is solved by scaling and processing of the input image in multiple resolutions – image pyramid. Embedded object detectors are often implemented directly in software using libraries such as OpenCV [21]. While this approach is easy and straightforward, it often is quite slow as detection is computationally demanding task and embedded processors tend

to be simpler and slower than desktop CPUs. Another approach is to implement a custom detection algorithm exploiting various acceleration resources of the target platform – CPU [22], GPU [23] or Field Programmable Gate Array (FPGA) [1, 20, 24, 25, 5, 4, 13] units. This is advantageous in many areas where the deployment of standard PC-based or embedded software solution is not possible, e.g. because of resource consumption, physical dimensions, industrial or military conditions, etc.

The object detection in embedded devices typically belongs to one of the three detection method categories. **1/** AdaBoost-based detectors – cascades of boosted classifiers [19] or soft cascades [26]. They typically use Haar image features [1, 2, 5, 4], or LBP [20]. **2/** Support Vector Machines (SVM) with Histograms of Oriented Gradient features (HOG) [27, 24, 28, 29, 25]; and **3/** Other methods implementing detection with background subtraction [30], keypoints [31], neural networks [32], or custom detection algorithms [33]. Most works, including thin one, belong to the first category, we give the detailed review of them in Section 3.2.

In this work, is proposed a simple and easy to use building block for FPGA that solves the object detection using state of the art boosted soft cascade classifier. We focused on implementation of the detection algorithm in the FPGA that efficiently utilizes the hardware resources and provides high performance. To produce classifiers for our hardware we used an existing, previously published algorithm [11]. The solution is multi-scale so it can detect objects of wide range of sizes.

It is suitable for various industrial applications, such as license plate detection, face detection, etc. The classifiers we use are especially suitable for hardware implementation since they are based only on pixel comparisons, look-up tables and integer-only calculations. Our architecture is extensively configurable, and it offers high image throughput even with high resolution inputs. In our main applications, which are detection of faces and license plates, we use processing of HD images (1280×720 pixels), but we also

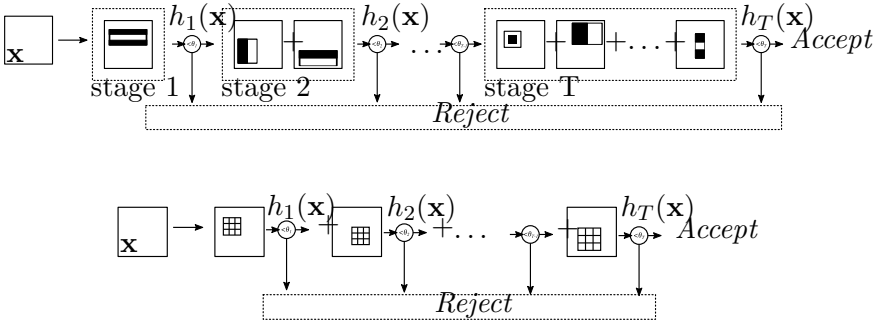

Figure 3.1: Comparison of Haar Cascade detector model (top) and Soft Cascade (bottom) that we use in our architecture. The main difference is that Soft Cascade does not contain *stages* and accumulates the response throughout the classifier. Another difference is that in Soft Cascade case the evaluation of the response can be terminated after every weak classifier.

present a configurations for processing images with resolutions up to 4K (UHD, 3840×2160 pixels). IP Core for face detection and other resources are available online. Our contributions are specifically:

- Advanced memory architecture for image representation in block RAM (BRAM) which allows for simple and fast data random access suitable for fast feature extraction.
- Cascading of detector blocks which allows for increasing the input image resolution and the total performance.
- Multi-scale object detection directly in FPGA enabled by cascading od SMEs, without using external components
- Re-usable detector block that can be easily incorporated into other architectures using standard interfaces.
- Efficient streaming and pipelining and advanced control that fully utilizes the engine resources.

3.1 Detector model

Let us first describe a framework for object detection that sliding window-based methods have in common [19, 27, 11, 34]. We assume the input image $\mathbf{I}$ to be a grayscale raster and a classifier $H(\mathbf{x})$ a function that accepts or rejects the image patch $\mathbf{x}$ and returns a confidence estimation.

Detection on a single image

The detection function $D(\mathbf{I}, H, a)$ classifies every fixed-size patch of the input image $\mathbf{I}$ by the classifier H. A patch is defined by its location (m, n). Its size (u, v) is fixed, defined during classifier training stage. We use $\mathbf{x} = \mathbf{I}(m, n, u, v)$ for patch extraction from the location (m, n). The detection function (3.1) returns the set of locations accepted by the classifier and scaled by factor a, and the classification confidence.

$$D(\mathbf{I}, H, a) \in \{([m, n, u, v] \cdot a, H(\mathbf{x}))\} \tag{3.1}$$

Multi-scale detection

The detection process is illustrated in Figure 3.2. From the input image $\mathbf{I}$, a pyramidal structure $\mathcal{I}$ (see Equation (3.2)) with k scaled versions is created, such that $\mathbf{I}_j$ is $\mathbf{I}_{j-1}$ downscaled by factor $S < 1$. In our architecture, we use $S = \frac{5}{6}$, which results approximately in a pyramidal representation with 4 scales per octave.

$$\mathcal{I} = \{\mathbf{I}_0, \mathbf{I}_1, \dots \mathbf{I}_{k-1}\} \tag{3.2}$$

The scale of j-th image in $\mathcal{I}$ can be retrieved as $s_j = S^j$; therefore, $\mathbf{I}_0$ corresponds to the original image. The result of the detection on $\mathcal{I}$, see Equation (3.3), is simply union of the results on individual images.

$$D(\mathcal{I}, H, S) = \bigcup_j D(\mathbf{I}_j, H, S^j) \tag{3.3}$$

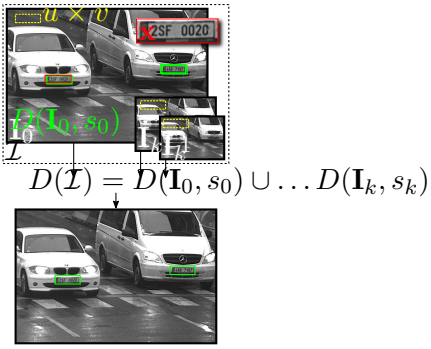

Figure 3.2: **(top)** The detection process on pyramidal image representation $\mathcal{I}$. The detection window of size u×v (yellow), is used for classification of every position m, n in image (example in red). The result of detection on each image is a set of locations $D(\mathbf{I}, s)$ accepted by the classifier (green). **(bottom)** The final detection result after non-maxima suppression.

The set $D(\mathcal{I}, H, S)$ is then processed by a non-maxima suppression (NMS) algorithm to suppress nearby detections and produce the final results for the image. We use a simple, overlap-based, NMS algorithm [34] which finds clusters of overlapping detections and keeps only the strongest detection from each cluster. However, other algorithms, such as mean-shift [27], could be used as well.

Table 3.1: Overview of state-of-the-art architectures. In *Length* column we report the number of features and number of stages for Cascades in brackets. *(self-organizing map neural network)

	Year	Image size [px]	Method	Feature type	Window size	Length	Scales	Object	Description
Lai [14]	2007	640×480	Cascade	Haar	20×20	52 (3)	15	Faces	Parallel calculation of feature responses
Zemcik [8]	2007	640×480	WaldBoost	LBP/LRD	31×31	80	-	Faces	Microprogrammable engine for feature extraction
Granat [2]	2007	256×256	AdaBoost	Haar	24×24	184	-	Faces	FPGA coprocessor for DSP
Cho [1]	2008	640×480	Cascade	Haar	20×20	2 135 (22)	18	Faces	Up to 3 features per clock cycle
Hiromoto [3]	2008	640×480	Cascade	Haar	24×24	2 913 (25)	18	Faces	Parallel calculation of feature responses
Martelli [25]	2011	640×480	SVM	Covariance	128×64	–	–	Peds.	Features extracted from 5x5 blocks
Kyrkou [4]	2011	320×240	Cascade	Haar	24×24	2 913 (25)	8	Faces	Combination of detector upscale and image downscale
Huang [5]	2011	320×240	Cascade	Haar	20×20	2 135 (22)	12	Faces	Scalable performance/resources tradeoff
Brouss [6]	2012	320×240	Cascade	Haar	20×20	2 135 (22)	15	Faces	Evaluation of multiple position in parallel
Jin [7]	2012	640×480	Cascade	LBP	20×20	250 (5)	16	Faces	Synthetic classifier
Kadlcek [13]	2013	1024×1024	WaldBoost	LBP	24×24	20	–	Faces	Synthetic classifier
Zemcik [20]	2013	640×480	WaldBoost	LBP/LRD	24×24	128	4	Faces	Programmable engine evaluating multiple positions
Yazawa [35]	2015	640×480	AdaBoost	HOG	40×80	–	5	Peds.,Cars	Featurtes extracted form blocks
Ma [36]	2015	1620×1200	SVM	HOG	64×128	3 708	34	Peds.	Featurtes extracted form blocks
Said [37]	2016	640×480	SVM	HOG	64×128	2 205	–	Peds.	Featurtes extracted form blocks
Kyrkou [28]	2016	800×640	SVM Cascade	LBP	20×20	1 062	18	Faces	Neural net pre-filter
Xu [38]	2016	320×240	Cascade	Haar	20×20	2 135 (22)	8	Faces	Features in FPGA, detector in ARM
Bilal [39]	2017	640×480	SVM	HOG	64×128	–	8	Peds.	Featurtes extracted form blocks
Yang [40]	2017	256×256	AdaBoost,SOM*	LBP	16×24	600	–	Faces	Heterogeneous parallel processor
Proposed	**2017**	**up to 4K**	**WaldBoost**	**LBP/LRD**	**up to N×27**	**1024**	**as required**	**Faces, LP**	Programmable engine with parallel processing of windows

The main work in the detection process is done by the classifier H which scores the individual image windows. The classifier cascade introduced by Viola [19] (or sometimes called Haar cascade), is a widely used model, see Figure 3.1. The classifier cascade analyzes the input image patch $\mathbf{x}$ by a sequence of progressively more complex *stages* composed from *weak classifiers* based on simple image *features*. After evaluating of a stage, the image patch can be either rejected (classified as background) or passed to the subsequent stage. The soft cascade, shown in Figure 3.1, is not explicitly divided into stages and the rejection decision is made after evaluating each weak classifier. In this work we use the soft cascade model based on Local Binary Patterns (LBP) or Local Rank Differences (LRD) features and we describe this model in detail in Section 3.4.

3.2 Related work

Current cutting edge object detection algorithms are based on deep learning and convolutional neural networks (CNN). Generally, they achieve high detection accuracy in comparison to linear classifiers (such as Adaboost or SVM) [41, 42]. On the other hand, the computation of convolutional layers is very demanding; the number of operations required for evaluation is several orders of magnitude higher compared to linear classifiers. Furthermore, the neural networks usually require large amount of intermediate results, increasing memory requirements during inference. Another issue is the number of network parameters which can easily reach many milions. Furthermore, the memory requirements of CNN-based detectors are prohibitive for FPGA implementation. Current state-of-the-art FPGA architectures is that why can process only small images [43] and they are very slow [44],or they must use clusters of very large and expensive FPGAs[45]. For these reasons, linear classifiers are still favorable for implementa-

tion in FPGAs and embedded devices in general especially when processing of large images is required.

Table 3.1 summarizes important works in the field of embedded object detection from last ten years. Here we analyze the approaches the authors used.

Lai et al. [14] proposed a parallel hardware architecture based on Haar cascades. They achieved a detection speed up to 143 frames per second (FPS) at VGA resolution. Due to high demands on FPGA resources they limited the cascade to only first three stages (52 features), which led to low detection accuracy. Their implementation is therefore suitable as a preprocessing unit rather than full object detector. Cho et al. [1] implemented a Haar cascade-based face detection algorithm. They implemented various versions with one or three parallel classifiers to accelerate the processing speed. The disadvantage is high memory demand to perform multiscale detection on a pyramid of integral images.

Huang and Vahid [5] developed a method to generate a Haar feature-based object detectors. They aimed at automatic generation of detectors with a required precision for FPGAs of various sizes. This approach allowed to reduce resource requirements of integral image memory and hardware complexity against universal implementation of detector. Brousseau and Rose [6] improved Haar cascade-based detector in FPGA by preloading of neighboring pixels, allowing parallel evaluation of classifiers in adjacent scanning windows. They also proposed a very complex evaluation control mechanism, allowing to rearrange execution of classifiers to coalesce the memory accesses.

Zemcik et al. [20], proposed an approach based on WaldBoost detection algorithm with LBP or LRD features. This approach implements stripe memory with block readout and image scaling but it is limited by fixed performace and by small image resolution, if the multi-scale detection is required.

Several authors proposed detection engines based on massive parallel execution of large number of features, increasing overall

performance at the expense of the resource consumption. Jin et al. [7] proposed a design of fully pipelined classifier for high-speed face detection with LBP cascades. The features in each stage are executed in parallel. Kadlcek and Fucik [13] proposed an automatic classifier synthesis for the FPGA. Their method generates a fast image preprocessing unit with LBP features, processing complete detection window per clock cycle. High expense of FPGA resources allows for implementation of only limited number of weak classifiers.

Most of the works implement AdaBoost Cascade of classifiers with Haar features for face detection [1, 14, 5]. But, for example Kyrkou [24] detected traffic signs and cars. Some authors solve pedestrian detection with SVM[35, 37, 36, 39].

3.3 Design choices

In this section, we analyze significant works from the point of efficient hardware implementation and we summarize the outlines for the design of our architecture.

When it comes to hardware implementation, Haar features are not a good choice for several reasons. Haar feature is evaluated as a convolution of image and a mask. Each feature in the detector can cover a different and potentially large number of pixels, which means many memory accesses. Without using an integral image, this cannot be implemented to run in constant time, which is an important feature for pipelining in hardware. Using of integral image increases memory requirements as each pixel requires higher bit depth [1, 14, 5, 24]. When using integral image, each feature can be evaluated by referencing from 6 to 9 pixels, depending on the shape of the feature [1]. Reading these values from BRAM, unfortunately, means non-uniform memory access which cannot be executed in a single clock cycle; therefore, most of the works implement the sliding window as a register array with FIFO line buffers stored in BRAM. This allows for parallel access of pix-

els in the window and evaluation of multiple features in parallel. However, this also leads into a huge multiplexer network (20×20 search window requires 400:1 multiplexer [1, 14]), that occupies many resources in FPGA. The resource consumption increases dramatically with the size of the detection window and thus such architectures are constrained to use only small and fixed window sizes to save resources. Huang [5] solves this drawback by limiting feature positions and simplifying the multiplexers.

Zemcik [20] substitutes Haar features with LBP which replaces shift registers and delay lines by a set of BRAM memory blocks with organization that allowed for the multiplexing to be replaced by a simple block addressing technique. This approach has another advantage in pipelining of feature evaluation. It allows simultaneous processing of multiple image windows in the stream and thus full utilization of the pipeline, which is not possible with standard scanning window approach [5, 2]. In general, the hardware detectors based on LBP features[7, 13, 28] achieves higher performance than Haar feature based detectors which is summarized in section 3.6.

Multi-scale detection is, in most cases, solved by storing the input image in RAM and scaling by an algorithm or circuitry independent on the detection unit [5]. Downscaled images are then passed to the detector from RAM one after another. Brouss [6] uses resolution so small that the image fits BRAMs in the FPGA. Kyrkou [24] combines image downscaling to half resolution and upscaling the detector window. Scaled version of the image is stored in BRAM. Granat et al. [2] scales the image features in the classifier and addresses the integral image at its original scale. Zemcik [20] scales image on the fly and stores only a narrow image stripe in BRAM. Some works [25, 13, 37] do not solve multi-scale detection and detects objects of a fixed size; therefore, their architectures are more simple and exhibits apparently higher performance.

As a basic building block in our architecture, we use an improved architecture by Zemcik et al. [20]. Specifically we improved the performance of pipelining, image scaling algorithm, the bit depth of the image and we extended it with cascading capabilites, described below. Our architecture differs from the others in several aspects. We use *soft cascade* instead of cascade of classifiers(see in Section 3.1). Soft cascade is usually more efficient in terms of the number of extracted features [34]. We use features that do not need integral image and that can be evaluated directly from the input image – LBP and LRD [46].

In our approach, the sliding window is not stored in FPGA registers. Instead, *Stripe Memory Engine* (SME) is used to store a narrow stripe of the input image in BRAM, see in 3.5. The stripe must be higher than the of classifier window (we use classifiers with height up to 24 pixels and stripe height is 32 pixels). In the classifier window, we limit geometric size of the features to 6×6 pixels which allows uniform reading of a fixed size pixel blocks from SME in one clock cycle. Juranek et al.[47] shows that limitation of feature block size does not have adverse effects on detector accuracy. Image is represented on 8 bits per pixel which saves resources compared to integral image where even 20 and more bits per pixel need to be used [24, 14]. The detector size is limited only by the height of the detection window but not by the width, which can be of virtually any size. We also do not use RAM to store the input image; instead, the image is scanned as it comes from the source and its scaled versions are generated on the fly (see in 3.5). Many image scales are stored in the same SME. The stripe memory, due to its organization, allows the evaluation of multiple scanning windows simultaneously and enables efficient pipelining and scheduling of the detector evaluation process. Moreover, dual-port BRAM allows us to implement two pipelines and therefore up to two features can be extracted in a clock cycle.

Another contribution of this work is that our detection engine can be cascaded in order to increase the performance and the im-

age resolution using *Stripe Memory Cascades* (see in 3.5). It is basically a chain of SMEs where one SME sends the image data to the subsequent one. The number of instances is only limited by available resources. In practical setup, one instance can hold few high-resolution image scales and the other the rest low-resolution scales; therefore, the maximum image resolution is bigger compared to one instance solution. Moreover, both instances run in parallel and thersefore the performance is also increased. The number of instances in the cascade is limited only by available resources.

All of these differences – detector based on simple image features, image representation in SME, cascading and efficient pipelining – contribute to low resource requirements and overall performance of the proposed architecture.

3.4 Classifier model

The main part of the detection is the evaluation of the classifier $H(\mathbf{x})$ on image patches. It consists of the feature extraction and the classifier response accumulation, which we describe in the following text.

Feature extraction

Given an image patch $\mathbf{x}$, a feature extraction is a function $y = f(\mathbf{x}, \pi), y \in \mathbb{N}$ which extracts a value from $\mathbf{x}$ based on the parameters π. As a feature extraction function, we use Local Binary Patterns (LBP) with

$$\pi = (x, y, w, h)$$

or Local Rank Differences (LRD) [46] with

$$\pi = (x, y, w, h, a, b)$$

where x,y,w,h define the feature position and the size in the patch $\mathbf{x}$ and a,b are indices of two distinct cells in the LRD case.

The feature response $f(\mathbf{x}, \pi)$ is evaluated from values of 3×3 cells whose positions and sizes are defined by the parameters π. The cell values $\mathbf{c} = C(\mathbf{x}, x, y, w, h)$ are obtained as a sum of pixel values in the respective cell. The two feature types we use, LBP and LRD, differ in how the values $\mathbf{c}$ are processed.

Local Binary Patterns (LBP)

In general, LBP is based on comparison of pixels from a circular neighborhood to the central pixel and generating binary code [48], forming the feature output. Extended versions attempt to reduce the number of possible output values by rotating the resulting bit pattern or by restriction of the number of 0-1 and 1-0 transitions in the code [49].

In this work, we use a simplistic variant of LBP which takes 3×3 cell values and generate 8 bit code form comparison of the central cell to the border cells. Mathematically, the calculation can be written as Equation (3.4) where $>$ operator compares all values of a vector to a scalar value, resulting in binary vector. Weights $\mathbf{w}$ correspond to powers of two

$$w = [1, 2, 4, 8, 0, 16, 32, 64, 128],$$

so the dot product effectively sets the bits in the result. The zero weight, $\mathbf{w}_5$, corresponds to the central cell $\mathbf{c}_5$ which is used as a basis for the comparison. The range of the resulting values of $\mathrm{lbp}(\mathbf{c})$ is $[0; 255]$.

$$\mathrm{lbp}(\mathbf{c}) = (\mathbf{c} > \mathbf{c}_5)\mathbf{w}^\top \tag{3.4}$$

Equation (3.5) shows how the feature value is calculated, given an image patch $\mathbf{x}$ and parameters π.

$$f(\mathbf{x}, \pi) = \mathrm{lbp}(C(\mathbf{x}, x, y, w, h)) \tag{3.5}$$

Local Rank Differences (LRD)

Features based on local ranks proved to be successful in object detection tasks [46]. LRD uses scheme similar to LBP – processing

of 9 values in 3×3 cells. It calculates the ranks of two distinct cells and outputs their difference. Mathematically, the function can be described as Equation (3.6), where a and b are indices of two distinct cells. The resulting value of the $\mathrm{lrd}(\mathbf{c}, a, b)$ values is in $[-8; +8]$ range.

$$\mathrm{lrd}(\mathbf{c}, a, b) = \sum \mathbf{c} > \mathbf{c}_a - \sum \mathbf{c} > \mathbf{c}_b \tag{3.6}$$

Equation (3.7) shows how the feature value is calculated, given an image patch $\mathbf{x}$ and parameters π.

$$f(\mathbf{x}, \pi) = \mathrm{lrd}(C(\mathbf{x}, x, y, w, h), a, b) \tag{3.7}$$

The classifier

A classifier H is represented as a sequence of T weak classification functions

$$h_i = (\pi_i, \theta_i, \mathbf{a}_i), \quad i \in 1, 2, \ldots T \tag{3.8}$$

where π are parameters for feature extraction, θ rejection threshold, and $\mathbf{a}$ look-up tables with response values. Given an image patch $\mathbf{x}$, the response of the classifier of length t, Equation (3.9), is a sum of predictions produced by the individual weak classifiers.

$$H_t(\mathbf{x}) = \sum_{i=1}^{t} \mathbf{a}_i(f_i(\mathbf{x}, \pi_i)) \tag{3.9}$$

The sample $\mathbf{x}$ can be rejected (classified as background) after evaluating $k < T$ weak classifiers when $H_k(\mathbf{x}) < \theta_k$. And it is classified as detected object only if all T weak classifiers were evaluated. $H_T(\mathbf{x})$ is then used as classification confidence. The evaluation is summarized in Algorithm 1.

The number of features evaluated on a sample is, therefore, not fixed as each image patch can be rejected by different number of weak classifiers. The number of weak classifiers varies depending on the image patch content. We can statistically evaluate the

Algorithm 1 Evaluation of classifier H on the sample $\mathbf{x}$.

procedure $H(\mathbf{x})$
  $h = 0$
  for $t \leftarrow 1, T$ **do**
    $(x, y, w, h, a, b) = \pi_t$ ▷ Decode parameters
    $\mathbf{c} = C(\mathbf{x}, x, y, w, h)$ ▷ Extract cells
    $g = \mathrm{lrd}(\mathbf{c}, a, b)$ ▷ Or $g = \mathrm{lbp}(\mathbf{c})$
    $H = H + \mathbf{a}_t(g)$ ▷ Accum. the confidence
    if $H < \theta_t$ **then**
      return $('reject', 0)$ ▷ Reject $\mathbf{x}$
  return $('accept', H)$ ▷ Accept $\mathbf{x}$

average number of weak classifiers required for classification of a patch – $\bar{t}$. The value can be viewed as computational effort required for classifier evaluation. It can be calculated on a dataset using (3.10) by counting the number of evaluated weak classifiers W and classified image patches P.

$$\bar{t} = \frac{W}{P} \tag{3.10}$$

The value largely depends on the task and training data. Usual values values are $2 < \bar{t} < 5$ [47]. Lower values means faster detectors. For illustration purposes, later in this paper, we use $\bar{t} = 2.5$ which is a realistic value e.g. for face detection [47]. The value is especially important since it directly influences the performance of the proposed architecture, see Section 3.5.

In practise, the classifier of length T with LBP features is represented by three matrices $\mathbf{F}$, $\mathbf{A}$ and $\mathbf{T}$, where F is $4 \times T$ matrix with feature extraction parameters $\pi_t = (x, y, w, h)$, $\mathbf{A}$ is $256 \times T$ matrix with lookup tables $\mathbf{a}_t$, and $\mathbf{T}$ is $1 \times T$ matrix with rejection thresholds θ_t for each weak classifier. The t-th column of the matrices correspond to parameters h_t. Note that in the case of LRD features, the size of $\mathbf{F}$ is $6 \times T$ and the size of $\mathbf{A}$ is $17 \times T$, since LRD has six parameters $\pi_t = (x, y, w, h, a, b)$ and 17 output

values for indexing. In Section 3.5 we use matrix **F** as a part micro program of the detection engine **A** and **T** are stored as lookup tables.

Classifier training

Detectors in this work are trained by WaldBoost algorithm [11]. But other algorithm producing a sequence of feature parameters and associates them with the corresponding response values can be used as well, e.g. [26]. The detailed description of the training algorithm is out of the scope of this paper since we focus mainly on the hardware implementation of the detection process. We kindly refer reader to the original paper [11]. Here we only provide informal description of the algorithm for reader to understand how it works.

The input of the algorithm is a pool of feature parameters, target *false negative rate* α, and a large set of training instances. E.g., when training a face detector, the training instances are image patches representing faces. The parameter α represents tradeoff between the final detector speed and its accuracy. Higher values of α (e.g. $\alpha = 0.2$) produces fast detectors with low value of $\bar{t}$, since they can reject background samples more rapidly. Low values (e.g. $\alpha = 0.01$) produces slower detectors with higher $\bar{t}$. We analyze this tradeoff in Section 3.6 on the task of face detection.

The training algorithm works in rounds, training weak classifiers one by one in s greedy manner. On the beginning of a round t, the algorithm loads background samples from a large set of images (not containing the target patterns) using the already trained classifier (i.e. weak classifiers from h_1 to h_{t-1}). For each feature in the pool, weak learner trains confidence values in lookup tables using AdaBoost [50]. In this step, the values are quantized to the resolution required by the FPGA. This is better than ex-post quantization (after the classifier is trained) since it allows training algorithm to adapt on errors caused by the computation with reduced precision [20, 8]. Then, the weak classifier minimiz-

ing exponential loss function [50, 19] is selected as h_t. Based on the distribution of H_t, for target and background samples, θ_t is selected so that as much as possible background samples may be rejected while discarding as few as possible target samples for the next round and satisfying target *false negative rate* α.

For the detector training in this work, we use our custom training software which produces detectors suitable for hardware, taking into account all possible quantization effects of the input image and values in lookup tables.

3.5 The architecture

We propose a hardware architecture that implements the key steps of sliding window object detection – image scaling, feature extraction, and classification of image patches. In the following text, we describe the design of the detector and its interface, and compare it to the equivalent software implementation in order to validate it. Figure 3.5 shows the overall schematic description of the detector.

Stripe Memory

The key part of our architecture is a *Stripe Memory Engine* (SME) which stores the active part of the input image and its scaled variants in multiple BRAMs, see Figure 3.5 for an illustration. When a new line is read from the image source, the data in SME are updated and scaled on the fly. The number of scales stored in SME is limited by the total width of SME raster, which is 4 096 pixels in this paper.

The architecture of SME is optimized for reading a block of pixels in a single clock cycle, so all data required for feature extraction are available in *constant* time. The data access is done in two stages. First, a fixed-size block aligned to certain position is retrieved from BRAMs to registers. Then, from this intermediate block, a sub-block with any size and alignment is retrieved by

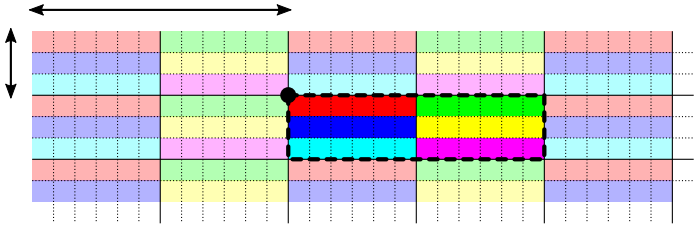

Figure 3.3: An example of SME in 2×3 organization and 6px blocks ($U = 2$, $V = 3$ and $B = 6$) stored in 6 BRAMS (color coded). Aligned block A of size 12×3 pixels can be retrieved from the memory in one clock cycle.

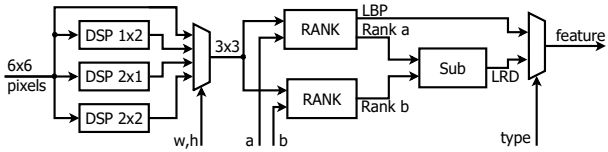

Figure 3.4: Circuit for feature extraction. The input is 6×6 pixel block from which, depending on feature parameters w, h, 3×3 cells are extracted. The resulting cells are used to calculate LRD or LBP features.

simple addressing. We store the image stripe in multiple BRAMs organized in a way that each BRAM is referenced only once when reading an aligned, fixed size block of pixels. BRAMs create a pattern of size U×V, and each BRAM stores B pixel block. This is illustrated in Figure 3.3 for U=2, V=3 and B=6. This requires $U \cdot V$ BRAMs to store the image stripe. Such an organization allows for reading $B \cdot U \times V$ pixel blocks (aligned to B pixels horizontally) in a single clock cycle by referencing all BRAMs.

Although SME can be configured almost arbitrarily, it is limited by the size of BRAM in the target platform. For practical applications, we use 4096×32 pixel raster, $U = 4$, $V = 8$, $B = 4$ and pixels represented on up to 9 bits. On our target FPGA, the

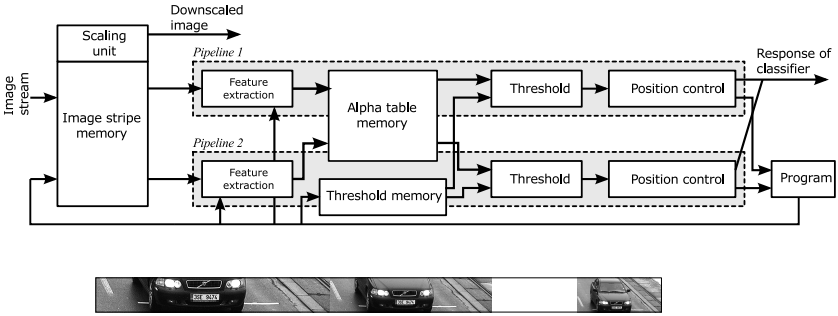

Figure 3.5: **(Top)** Block diagram of detector. The SME unit, which stores the image and produces downscaled images and two detection pipelines, driven by microcode program. **(Bottom)** Illustration of the stripe memory that we use for image representation and source of data for detector evaluation is shown, too. Incoming line (blue) is stored as a last line in the buffer. When possible, 6×6 blocks on the bottom of the buffer are scaled and stored as 5×5 blocks in subsequent scales. See the text and Figure 3.3 for details on how the data are stored in FPGA BRAMs.

SME takes 32 BRAMs with 36 kbit capacity. This organization allows us to retrieve 16×8 pixel blocks aligned to 4 pixel position. The, for feature extraction, we take 6×6 pixel sub-block or 8×8 sub-block for image scaling.

Feature extraction unit

The detection engine implements LBP and LRD image features. The size of the feature cells is limited to $w \leq 2, h \leq 2$, and thus the feature area is limited to a maximum 6×6 pixels. The position x, y is not limited in any way.

The block diagram of feature extractor is shown in Figure 3.4. The input is the 6×6 pixel block read from SME according to the absolute feature position in image (taking into account position

of analyzed window). DSP blocks extracts all possible variants of $\mathbf{c}$ from the SME. One of the variants is selected for evaluation according to the feature parameters w, h from π. The ranks of elements a and b are calculated as the number of positive comparisons of an element $\mathbf{c}_a$ (resp. $\mathbf{c}_b$) to all other elements in $\mathbf{c}$. The ranks are subtracted to obtain the LRD feature value. Evaluation of an LBP feature is similar – parallel comparison of the central element $\mathbf{c}_{\mathbf{1},\mathbf{1}}$ to the elements at the boundary. The response of a weak classifier is obtained from the look-up table $\mathbf{a}$ associated with the extracted feature.

It should be noted that the circuit is designed to extract both LRD and LBP features simultaneously; however, in case that only one feature type is used, the circuitry for the other type is optimized-out during synthesis.

Detector control

The detector implements Algorithm 1. For every position (m, n) in SME image a sequence of instructions is executed. Each instruction reads the 6×6 pixel block from SME, extracts 3×3 cells $\mathbf{c}$, evaluates the feature function and accumulates the response value read from table $\mathbf{A}$. Then, the window is rejected or passed to the next stage based on the threshold value from $\mathbf{T}$. Everything is driven by the parameters from the instruction code. In case of rejection, new position is scheduled for evaluation. When all the instructions are finished, the window coordinates and the confidence value are sent to output.

The detector itself is controlled by a programmable automaton driven by a 32-bit instruction set. An instruction encode parameters for feature extraction – particularly the feature parameters from matrix $\mathbf{F}$ and sequence number identifier t for addressing matrices $\mathbf{A}$ and $\mathbf{T}$. We use 8 bits to encode each coordinate of feature position (x, y), 2 bits for (w, h), and 4 bits for each from (a, b). Note, that values a and b are present in the instruction code even for LBP-based detectors where they are unused. In the matrix

A, we store the response values on 9 bits and the thresholds in **T** table on 18 bits.

The detector microcode contains a sequence of up to 1 024 instructions; it means the length of the classifier is $T \leq 1024$. The number of instructions can be decreased or even increased, having linear impact on the memory requirements. Current implementation requires 1 BRAM for storing instructions, 1 BRAM for thresholds and 5 BRAMs for **A** and **T** in LRD case (64 BRAMs in LBP case) 2 BRAMs are occupied by instructions for static execution scheduler (see 3.5).

The evaluation of the classifier is pipelined. The pipeline is 14 clock cycles long and thus up to 14 positions are evaluated simultaneously. Thanks to the memory architecture described above, the pipeline can be utilized to 100 % which is impossible to achieve by previous scanning window approaches [5, 2]. We use two-port BRAM in SME, so we use two pipelines to double the performance. However, a small portion of memory accesses from the second pipeline needs to be reserved for image scaling and for storing the incoming image lines and the down-scaled data back to the SME – we leave one out of every 4 clock cycles for the scaling unit to generate the scaled images, and therefore the overall performance is $\bar{p} = 1.75$ features extracted per clock cycle.

Image scaling

Besides the original image, SME stores scaled variants of the image. The scaling is done on-the-fly over few last image lines. We use a block-based approach for scaling with fixed factor $S = \frac{5}{6}$, where 6×6 pixels blocks from a base scale are transformed to 5×5 pixels blocks in the subsequent scale. We implemented the separable, integer version of Lanczos [51] scaling algorithm for 8 bit images.

The process of downscaling and detector execution on individual SME lines is statically scheduled and driven by the microcode stored in BRAM. The classifier operations are performed to ev-

ery line but every scale has a different number of lines to process. Moreover, the scaling is a block operation which is performed every 6-th line. This can cause occasional bursts of detector executions. The static scheduling allows us to distribute the execution of detector and scaling to avoid this execution bursts and ensure regular processing of image stream.

The maximum height of scanning window is given by height of SME minus size of block produced by scaling unit, which is 27px (32-5px in our case). This height is sufficient for many of detection tasks and also standard detectors uses similar dimensions – 21×21 or 24×24 pixels [19, 11].

Detector interface

From the outside view, the detector is a computational block with one input, one configuration interface and two outputs. The input reads a stream of incoming image data of the given resolution. The configuration interface itself is composed from the detector definition (instructions and associated look-up tables), input image size, and sizes of scaled versions. The first output is a stream of the image data from the smallest scale in the SME. This output is used as an input for another detector instance. The second output contains detection results – coordinates and scale of detected objects. For both image input and output, we use AXI Stream Video interface for configuration the AXI-Lite interface and AXI Stream for detection results. This interfaces simplify integration of the detector to applications.

Stripe memory cascades

A single detector block is limited by the width of SME (4 096 pixels in our case) and by the performance of feature extraction, which is $\bar{p} = 1.75$ features/Hz (i.e. 350 M features/s with 200 MHz clock). From the performance point of view, it is not efficient to build the detector with a wider window (buffer) to hold more image

scales, because the limitation of feature extraction speed would still remain. Our design allows for a more efficient solution – cascaded connection of detector blocks which we call *Stripe Memory Cascade*, illustrated in Figure 3.6. In the cascade, one detector instance generates scaled version of image and passes it to the subsequent instance. No limitation exists on the number of instances, except for the resources available on the target platform. All instances operates in simulataneously, effectively increasing the speed of the feature extraction. Output streams from all the instances are simply merged to form the output of the cascade.

Table 3.2 shows several configurations of cascaded instances, their performance, and resources they require. The naming convention we use for the configurations encodes the resolution processed and the number of detector block instances, e.g. `VGA/1` is configuration for processing of VGA image with one detector block instance and it is similar to what was proposed by Zemcik et al. [20]. Versions `HD/2`, `HD/3` and `HD/4` are configurations for HD image with different performance and resource requirements due to different assignment of image scales to detector instances. Versions `FHD/4` and `UHD/7` are for Full HD and 4K images. The versions for LBP and LRD differs mainly in memory requirements because LBP requires more BRAMs for classifier definition as described earlier in this paper.

Speed analysis

The theoretical maximum throughput (in frames per second) for one instance of the detector unit can be estimated using Equation (3.11) where f is the operating frequency, $\bar{p}$ the number of features extracted in one clock cycle, $\bar{t}$ is the average number of weak classifiers evaluated per window, and P represents the number of positions to evaluate in the image and its scaled versions assigned to the detector. The numerator of Equation (3.11) represents the total number of features extracted by the detector, the denominator is the average number of features that must be

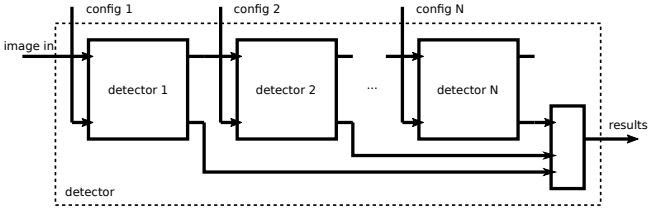

Figure 3.6: Cascading of detector instances. Each detector takes scaled input image from the previous one, the output coordinates and classifier response of detected objects are merged into one stream. Each detector is configured separately.

extracted on an image. As explained in Section 3.5, in our architecture has $\bar{p} = 1.75$. The value of $\bar{t}$ is the property the particular classifier, the average number of features that needs to be extracted from the image in order to decide the class of one analyzed window (see Section 3.4). It reflects the average case and it can change locally with irregularities in data that are hard to predict. We use $\bar{t} = 2.5$ for illustration purposes which is a realistic value for face detection. See Section 3.6 with the analysis of detectors we use.

$$F = \frac{f \cdot \bar{p}}{P \cdot \bar{t}} \tag{3.11}$$

The throughput of the whole cascade of detectors is limited by the slowest unit in the chain and it depends largely on sizes of images processed by the individual instances. In the Table 3.3, we show breakdown of `HD/*` variants from Table 3.2). Each version processes 20 image scales, the difference is in the manner how the scales are assigned to the detectors in the chain, and in the length of the chain.

Let us focus, for example, on the `HD/2` version, with two instances of detector. The first instance contains four image levels (resolutions from 1 280 pixels to 742 pixels of width) and we estimate around 5.4 M features need to be calculated on those four levels on average. Therefore, the speed of the first instance is around

Table 3.2: Examples of cascade configurations, their predicted performances, and resource requirements. Valid for detector of size 21×21 px, $\bar{t} = 2.5$, and $f = 200\,\mathrm{MHz}$.

Version	Feature	Res. [pixels]	Scales	Insts.	BRAM	REG	LUT	FPS
`VGA/1`	LRD	640×480	18	1	41	7640	9933	160
`HD/2`	LRD	1280×720	20	2	82	15292	19919	64
`HD/3`	LRD	1280×720	20	3	123	22944	29905	94
`HD/4`	LRD	1280×720	20	4	164	30596	39891	159
`FHD/4`	LRD	1920×1080	22	4	164	30596	39891	60
`UHD/7`	LRD	3840×2160	26	7	288	53552	69849	17
`VGA/1`	LBP	640×480	18	1	100	7650	9978	160
`HD/2`	LBP	1280×720	20	2	200	15312	20009	64
`HD/3`	LBP	1280×720	20	3	300	22974	30040	94
`HD/4`	LBP	1280×720	20	4	400	30636	40071	159
`FHD/4`	LBP	1920×1080	22	4	400	30636	40071	60
`UHD/7`	LBP	3840×2160	26	7	700	53622	70164	17

64 FPS, calculated using Equation (3.11). The second instance contains the rest of the image scales (resolutions from 619 pixels to 42 pixels of width) and its speed is estimated to 233 FPS. The total speed of the `HD/2` is therefore 64 FPS as it is the minimal framerate from all detectors in the chain.

Validation

During the design phase, we developed a software implementation of the detection algorithm which uses the same input data as the hardware implementation (look-up tables, instructions, thresholds, etc), and is based on the same image scaling algorithm. The architecture was validated by comparison of the results produced by the software implementation to the results produced by our architecture on a large set of images. The results were identical; therefore, we assume that the subtle differences in implementation in software and hardware are acceptable.

Table 3.3: Comparison of three different cascade designs, all for HD resolution. The values are shown for detector of size 21×21 px with average $\bar{t} = 2.5$ features per position, and $f = 200\,\mathrm{MHz}$ clock. It can be observed that trade-off between the number of instances and desired detection performance exists.

Scale	Resolution	$P \cdot \bar{t}$	HD/2			HD/3			HD/4		
			$\sum P \cdot \bar{t}$	$\sum W$	FPS	$\sum P \cdot \bar{t}$	$\sum W$	FPS	$\sum P \cdot \bar{t}$	$\sum W$	FPS
1	1280×720	2200103	5468605	3979	64	3714188	2347	94	2200103	1280	159
2	1067×600	1514085							1514085	1067	231
3	890×500	1040628				3058318	2856	114	1754418	1632	199
4	742×417	713790									
5	619×348	488865	1497155	3536	233				1497155	3536	233
6	516×290	332887									
7	430×242	225972									
8	359×202	152945									
9	300×169	103230									
10	250×141	68700				193255	1312	1811			
11	209×118	45590									
...	...	...									
20	42×25	210									
FPS					**64**			**94**			**159**

3.6 Results and Evaluation

In our applications we use Xilinx Zynq SoC with ARM CPU and FPGA. This combination allows for simple configuration of the detector and post processing of the results. However, if required, everything can be fixed and implemented in FPGA only. The design was written completely in VHDL with only few platform-dependent blocks (such as 36 kbit BRAM capacity); thus, it could be relatively easily adapted to various FPGAs, even from different vendor.

We built a prototype of a smart camera with HD CMOS image sensor and Zynq SoC Z-7020 chip. The camera captures image at 60 FPS and passes it through the `HD/2` detector. The detection results are processed on ARM core (non-maxima suppression, filtering) and the image along with the coordinates of detected objects are streamed through the network. We demonstrate the architecture on the detection of frontal faces and detection of license plates. As an example of our technology, we provide an IP Core

of version `VGA/1` and `HD/2` detector with built-in face detector[1]. This IP Core takes approximately 15 % of Zynq Z-7020 resources.

Detector evaluation

Properties of WaldBoost detectors were experimentally evaluated many times on various problems [11, 52, 47, 53]. We tested our architecture in two example scenarios – face detection and license plate detection. These two applications are important in surveillance tasks. However, the detector can be used for detection of other rigid objects as well - cars [54], pedestrians [35] etc. We compare our detectors to the pre-trained detectors from OpenCV which implement Haar and LBP Cascades used by other state-of-the-art architectures. We report Receiver operating curve (ROC) – the tradeoff between *false positive rate* (the number of false detections generated per one image) and *miss rate* (the ratio of missed objects). Figure 3.9 shows a few images from each of the tasks.

Detection of frontal faces

We trained frontal face detectors on a large dataset of faces and compared them to OpenCV cascade detectors widely used by other authors as a baseline [1, 5, 6, 38]. The detector window size (u, v) was set to 24×24 pixels and the detector length to $T = 1024$. We trained four detectors with different target *false negative rate* $\alpha \in \{0.01, 0.05, 0.1, 0.2\}$, see Section 3.4 for details. From OpenCV, we used `haarcascade_frontalface_alt`, as it gives the best results from the built-in detectors. We tested the detectors on our set of 102 high resolution images with 1 857 annotated frontal faces (which is bigger and more challenging than MIT-CMU usually used for testing of frontal face detectors). The results in Figure 3.7 show that our detector (with $\alpha = 0.1$) gives almost 10× less false positives compared the detectors from OpenCV at the same recall

[1] All resources can be downloaded from https://github.com/RomanJuranek/zynq-detector

level. The *recall* of OpenCV detectors is 94 % as reported by others [1, 20, 24, 25, 5, 4, 13]. Table 3.4 summarizes the speed and recall tradeoffs of the detectors trained with different value of α and their predicted performance in FPS when executed in version `HD/2` architecture. 60 FPS margin is satisfied by classifiers with $\alpha \leq 0.1$.

Detection of license plates

In law enforcement applications, such as speed measurement, detection of licence plates is a crucial step where accuracy and speed matters very much. We trained a license plate detector on a proprietary database of images taken by speed enforcement cameras. The dataset contains 30 000 automatically obtained samples of axis aligned license plates. The test set contains 1 000 images with manually corrected annotations. The dataset covers a wide range of conditions – day, night, sun, rain, snow and fog. For our experiments, the detector window size (u, v) was set to 84×12 pixels and the detector length was $T = 1024$. Accuracy evaluation in Figure 3.7 shows that the detection rate of WaldBoost detector is over 99 % when a false alarm occurs on one out of one hundred images. Detector speed measured on the test set is $\bar{t} = 2.7$, corrensponding to 62 FPS in `HD/2`. This is more than sufficient for this kind of application. For comparison, we trained Haar and LBP cascade from OpenCV on the same data using tools installed with the library. As Figure 3.7 suggests, our detector outperforms OpenCV detectors by a large margin.

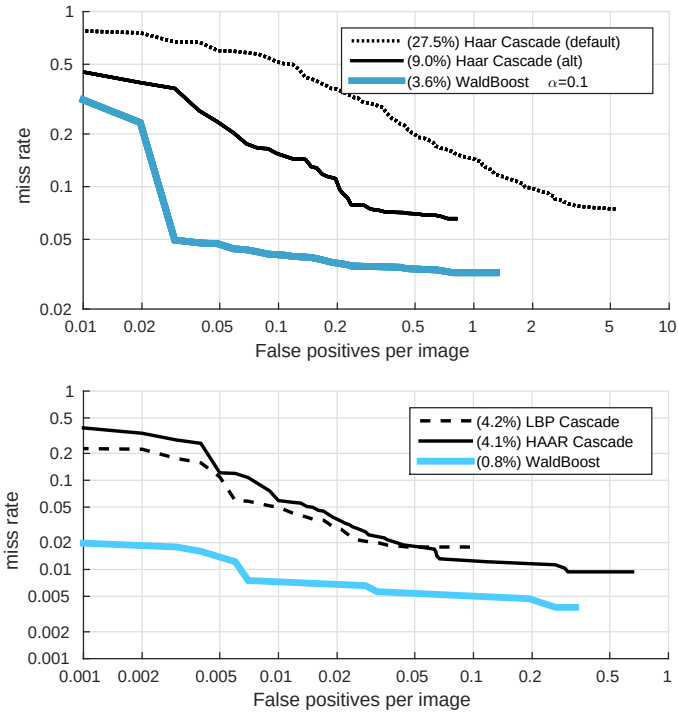

Figure 3.7: Accuracy evaluation of our detectors (WaldBoost) and comparison to OpenCV detectors (Haar and LBP cascade) for frontal face detection (top) and license plate detection (bottom). WaldBoost gives lower false positives at the same accuracy level.

Table 3.4: Speed analysis of our face detectors on `HD/2`. Fast detectors have slightly lower accuracy. The important value is $\bar{t}$, which directly influence the performance of our architecture.

	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.2$
Recall	0.970	0.969	0.964	0.952
$\bar{t}$	7.54	4.43	2.50	1.96
F [FPS]	21	36	64	82

Table 3.5: Power Consumption comparison. FPGA-based detector is more power efficient compared to PC and GPU solutions.

	platform	FPS	Power (W)	mJ / Frame
PC	Intel i7 3770K	22	77	3500
GPU	GeForce 1080Ti	915	120	131
SOC	Tegra K1	38	4	105
`HD/2`	Artix7 xc7a75	64	1.52	24
`HD/4`	Artix7 xc7a200	159	2.95	19

Table 3.6: Comparison of critical parameters of the published architectures reported by the authors to the proposed architecture. Some works do not report all parameters or report *slices* (SL) or *logical elements* (LE). *Suitable only for preprocessing purposes

	Feature	Platform	Res. [pixels]	FPS	Stride	*wpc*	*f* [MHz]	BRAM	LUT	REG	DSP
Lai* [14]	Haar	Virtex2 VP30	640×480	143	1	0.886	126	44	20 900	7 800	–
Granat [2]	Haar	Virtex2 LX250	256×256	< 5	1	0.058	24	100	–	–	–
Cho [1]	Haar	Virtex5 LX110T	640×480	7	1	–	–	41	66 900	21 900	–
Hiromoto [3]	Haar	Virtex5 LX330	640×480	30	1	0.173	160	–	63 440	55 515	–
Martelli [25]	Covariance	Virtex6 LX240T	640×480	132	8	0.002	154	3	1 553 (SL)	–	22
Kyrkou [4]	Haar	Virtex2 VP30	320×240	64	1	0.083	100	24	25 800	23 800	–
Huang [5]	Haar	Virtex5 LX155T	320×240	100	1	0.268	65	–	80 000	–	–
Brouss [6]	Haar	Stratix4 GX530	320×240	50	1	0.083	125	–	–	–	–
Jin [7]	LBP	Virtex5 LX330	640×480	300	1	1.974	125	286	128 041	74 997	–
Kadlcek* [13]	LBP	Virtex2 LX250	1024×1024	130	1	0.948	137	17	1 007 (SL)	–	–
Zemcik [20]	LBP/LRD	Spartan6 LX45T	640×480	160	1	0.726	152	31	7 373	1 732	–
Yazawa [35]	HOG	Cyclone III	640×480	13	5	0.002	70	–	17 419 (LE)	11 306	–
Ma [36]	HOG	Virtex-6 LX760	1620×1200	10	4	0.045	150	381	46 238	186 531	190
Said [37]	HOG	Virtex6 LX240T	640×480	292	4	0.017	222	8	1 357 (SL)	–	46
Kyrkou [28]	LBP	Spartan6 LX150T	800×600	40	1	0.827	70	256	32 532	20 153	59
Bilal [39]	HOG	Cyclone IV	640×480	25	4	0.015	50	3	751	496	0
Ours LRD (HD/2)	**LRD**	**Zynq Z-7020**	**1280×720**	**64**	**1**	**0.930**	**200**	**82**	**19 919**	**15 292**	**0**
Ours LBP (HD/2)	**LBP**	**Zynq Z-7045**	**1280×720**	**64**	**1**	**0.930**	**200**	**200**	**20 009**	**15 312**	**0**
Ours LRD (UHD/7)	**LRD**	**Zynq Z-7045**	**3840×2160**	**17**	**1**	**2.334**	**200**	**288**	**69 849**	**53 552**	**0**

Power Consumption Comparison

Table 3.5 shows estimation of power consumption of different platforms executing the face detection algorithm with Waldboost classifier ($\bar{t} = 2.5$, $\alpha = 0.1$) on 1280×720 images. On CPU, GPU, and Tegra, we used an OpenCL implementation of the detection algorithm from Herout et al. [23]. The critical steps were implemented in OpenCL and compiled for the target platform, efficiently exploiting SSE/AVX instructions and multi-thread execution on CPU and computing cores as well as texturing units on GPU. For the Intel CPU, we report maximum thermal design power (TDP). In case of GPU and SoC, the accurate chip power consumption is available. Power consumption of the FPGA was estimated using Xilinx Power Estimator, assuming the worst case with a 100% toggle rate (i.e. when signals flips every clock cycle). For the measurement purposes, we synthesized the `HD/2` in the different FPGA of the same family without the ARM core, so the results are not influenced by the power consumption of the ARM which is, in fact, not required during the detection. For all platforms, we report the metric which expresses energy consumed by the platform per one frame (Joules/Frame). The Table 3.5 shows that the FPGA design requires approximately five-times less energy than SoC Nvidia Tegra.

Comparison to other architectures

Table 3.1 shows the comparison to other architectures in terms of the maximum image resolution, detection algorithm and features, scanning window size, and type of detected object.

Due to our unique stream memory cascades, the detector can process images at very high resolution (up to 4K) while it is still capable of detection of very small objects. This property may be important e.g. when surveillance camera covers a large area. The most state-of-the-art architectures are capable of processing up to 1Mpix images.

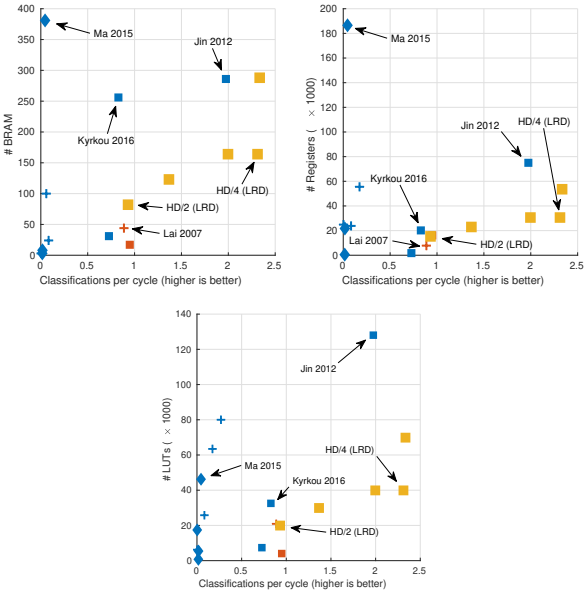

Figure 3.8: Comparison of classifications per cycle (*wpc*) and resource requirements of the architectures from Table 3.1. Yellow color encodes *proposed* architectures from this paper, red color *preprocessing units*, blue color the rest of the architectures. Marker shape encodes feature type used by the architecture (◆ stands for HOG, ✚ for Haar features, and ■ for LBP/LRD features).

The advantage of the proposed architecture, comparing to others, is the optional size of the detection window, which is limited only by the height of SME, while the width remains unlimited and freely adjustable. In other architectures [5, 2, 1, 6] the changing of window size means re-synthesizing of the whole design and, what is worse, larger windows takes more resources for multiplexer networks required for reading out the pixels. This is completely avoided in the proposed solution.

We compare our architecture to other similar ones. However, they are realized by different FPGA technology, have different input sizes, classifier strides in image and other parameters. To make comparison possible, we characterize all the architectures by number of processed scanning windows per clock cycle (*wpc*) which gives raw performance measure independent on the used technology, frame rate and other parameters. Table 3.6 summarizes performance and resource requirements of our architecture and compares it to the other published works. Plots in Figure 3.8 show the dependency of resource consumption on the *wpc* classification for some architectures. Various proposed configurations of cascade instances (from Table 3.2) are plotted in each graph. It can be observed that overall performance increases with number of cascaded SME instances which proves its benefit.

The resource consumption and performance of configuration `HD/2` is comparable to Kadlcek [13] and Lai [14], anyway, their design achieves only low detection accuracy caused by very short detector and limits their possible usage for preprocessing purposes only. Our architecture, on the other hand, works as fully featured object detector while providing sufficient performance even at high image resolutions. In summary, the graphs on Figure 3.8 and Table 3.6 shows the performance superiority of LBP/LRD feature based detectors to Haar and HOG based detectors.

The solution by Jin [7] and Kyrkou [28] achieves very high *wpc*, comparable to our `FHD/4` and `HD/2` configurations, but they require multiple-times more resources against our solution, even for only low image resolution. Comparing to work of Zemcik [20], the architecture we improved, we achieved higher performance, *wpc*, and maximum image resolution due to a cascading nature of our design.

SVM based classifiers using HOG features presented by Said [37] and Martelli [25] achieves high framerate mainly due to detection stride, where they process only every x-th image row and column, effectively making the image $x\times$ smaller, which reflects into low

Figure 3.9: Examples of detected objects on selected images from testing datasets for face detection (top) and license plate detection (bottom).

throughput. Moreover, they can only detect objects with fixed size 128×64 pixels as they do not solve multi-scale detection. This is why their architecture performs so well with so limited resources. However, to detect smaller objects, they need to upscale the image, and for multi scale detection, pyramidal representation need to be created, increasing the search space and slowing down the detection.

The proposed architecture achieves the highest performance (represented by *wpc*) compared to the others and also has a relatively low resource consumption as is evident from the Table 3.6 and graphs on the Figure 3.8.

Chapter 4

Conclusion

The main goal of the research conducted in this thesis was to deeply investigate methods for optimising the object detector in images running on FPGA. The newly proposed methods enable creating an object detector in hardware that outperformed state-of-the-art in better detection performance, better performance/resources ratio and better accuracy in selected application tasks. The object detector has been developed using FPGA solely and tested on the face and license plates detection.

The proposed detectors use boosted soft cascades of classifiers with local image feature as weak classifiers. The combination of the unique structure of memory and the local features enabled the effective sequential evaluation of weak classifiers. Parallel processing of multiple independent positions in the image significantly increased detection performance. Cascade connection allows to distribute the calculation among multiple detectors optimally and thus scale the performance and resources consumption to the specific application. The newly designed method enables an efficient multi-scale detection on-the-fly without the use of external memory and large FPGA memory requirements.

The presented scientific contribution aids to create an object-in-image hardware detector usable in practice. A significant benefit is the scalability of performance and resource consumption. It

rends possible to develop a detector, which can process either a FullHD video at 60 fps on FPGA with a current price of approximately 100 USD or an HD video at the same speed on a chip with a third of the price. It also allows to process images in high resolution; the proposed hardware detector is the first presented solution for the detection of all-sized objects, even tiny, at 4K resolution. In terms of accuracy, the proposed detector achieves better results than other similar detectors. On the comparative face detection dataset, it achieved 97% accuracy compared to only 91% being the best result until then.

The proposed detectors are expected to be utilised in smart cameras in industrial, transport or security applications, i.e. in tasks such as the detection of faces, pedestrians, products, licence plates. In this work, the practical use is demonstrated on the task of license plate detection for parking control in residential areas. The proposed solution has many advantages over the existing system, such as lower power consumption leading to decrease of heating, as well as lower cost and smaller size of the resulting system.

The use of boosted classifiers for object detection in hardware still remains a reasonable approach. Neural networks are the state-of-the-art in non-hardware object detection; however their deployment on FPGA has enormous resource requirements. In the future, I can see a benefit in combining boosted detectors with CNN to improve the accuracy of detection while maintaining reasonable demands on computing resources and the cost of devices. Another possible way to improve accuracy could be the use of multi-channel features such as ACF.

Curriculum Vitae

PETR MUSIL
Petr Musil received the M.Sc. degree from the Brno University of Technology. He is currently a member of the Graph@FIT Group, Department of Computer Graphics and Multimedia, Faculty of Information Technology, Brno University of Technology. His interests include image processing and computer vision algorithms and their acceleration on programmable hardware and embedded devices.

Education

- 2010 - 2012: Brno University of Technology, Faculty of Information Technology, **Master of Computer Graphics and Multimedia**
- 2007 - 2010: Brno University of Technology, Faculty of Information Technology, **Bachelor of Information Technology**

Publications

High Performance Architecture for Object Detection in Streamed Videos, conference paper
Pavel Zemčík, Roman Juránek, Martin Musil, Petr Musil and

Michal Hradiš, Proceedings of FPL 2013, ISBN 978-1-4799-0004-6, DOI: 10.1109/FPL.2013.6645559 **Author participation: 30%**

High performance FPGA object detector: Hardware prototype, conference demo
Pavel Zemčík, Roman Juránek, Martin Musil, Petr Musil and Michal Hradiš, Proceedings of FPL 2013, ISBN 978-1-4799-0004-6, DOI: 10.1109/FPL.2013.6645622 **Author participation: 30%**

High performance architecture for object detection in streamed video (abstract only), poster
Pavel Zemčík, Roman Juránek, Petr Musil, Martin Musil and Michal Hradiš, Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays, 2013
DOI: 10.1145/2435264.2435319 **Author participation: 20%**

Single-Loop Approach to 2-D Wavelet Lifting with JPEG 2000 Compatibility, conference paper
David Bařina, Martin Musil, Petr Musil and Pavel Zemčík, IEEE 27th International Symposium on Computer Architecture and High Performance Computing Workshops, ISBN 978-1-4673-8621-0, DOI: 10.1109/SBAC-PADW.2015.10 **Author participation: 30%**

High Dynamic Range Video; Concepts, Technologies and Applications, book chapter
Pavel Zemčík, Petr Musil and Martin Musil, Elsevier Science 2016, ISBN 978-0-12-809477-8 **Author participation: 33%**

Real-Time HDR Video Processing and Compression Using an FPGA, book chapter
Martin Musil, Petr Musil and Pavel Zemčík; Elsevier Science 2016, ISBN 978-0-08-101038-9, DOI: 10.1016/B978-0-12-809477-8.00007-8 **Author participation: 35%**

True HDR camera with bilateral filter based tone mapping, conference paper
Svetozár Nosko, Martin Musil, Petr Musil and Pavel Zemčík, SCCG '17: Spring Conference on Computer Graphics 2017, ISBN 978-1-4503-5107-2, DOI: 10.1145/3154353.3154367 **Author participation: 20%**

Cascaded Stripe Memory Engines for Multi-Scale Object Detection in FPGA, journal article in Scopus
Petr Musil, Roman Juránek, Martin Musil and Pavel Zemčík, IEEE Transactions on Circuits and Systems for Video Technology 2020; vol. 30, no. 1, pp. 267-280, Jan. 2020,
doi: 10.1109/TCSVT.2018.2886476. **Author participation: 30%**

Unconstrained License Plate Detec-tion in Hardware, conference paper
Petr Musil, Roman Juránek and Pavel Zemčík, International Conference on Vehicle Technology and Intelligent TransportSystems (VEHITS) 2021. **Author participation: 40%, Under review**

Projects

- **Moderní metody zpracování, analýzy a zobrazování multimediálních a 3D dat**, 2020–2022
- **Zpracování, zobrazování a analýza multimediálních a 3D dat**, 2017–2019
- **Zpracování, rozpoznávání a zobrazování multimediálních a 3D dat**, 2014–2016
- **Java platform for hIgh PErformance and Real-time large scale data management (JUNIPER)**, 2013–2015

- **7H12006, ConstRaint and Application driven Framework for Tailoring Embedded Real-time Systems**, 2012–2015

- **TE01020415, Centrum kompetence ve zpracování vizuálních informací**, 2012–2019

- **LD12027, Pořizování a zpracování HDR obrazů**, 2012–2015

Creative Activities

- AdaBoost in VHDL; Image detection using AdaBoost in VHDL; software; MUSIL, P.; MUSIL, M.; ZEMČÍK, P.; JURÁNEK, R.

- Zynq Profiler; Zynq Profiler; software; MUSIL, P.; MUSIL, M.; ZEMČÍK, P.

- AdvHDRVideoCam; Funkční vzorek zlepšené kamery pro snímání HDR videa; functional specimen; MUSIL, M.; MUSIL, P.; SEEMAN, M.; ZEMČÍK, P.

- HDRVPS; HDR Video Processing Software; software; MUSIL, P.; MUSIL, M.; SEEMAN, M.; ZEMČÍK, P.

- ForegroundDetDemo; Embedded detektor popředí v obraze; functional specimen; MUSIL, M.; MUSIL, P.; KOPLÍK, K.; ZEMČÍK, P.

- fpga-dwt; FPGA Cores for Discrete Wavelet Transform; software; MUSIL, P.; MUSIL, M.; BAŘINA, D.; ZEMČÍK, P

- WaldBoostFPGA; Funkční vzorek Zařízení pro detekci v obraze pomocí WaldBoost v FPGA; functional specimen; MUSIL, P.; MUSIL, M.; ZEMČÍK, P.

- DetectorCore; VHDL object detector - IP core; functional specimen; MUSIL, P.; MUSIL, M.; ZEMČÍK, P.; JURÁNEK, R.
- ZynqHDRVideoCam; HDR kamera; functional specimen; NOSKO, S.; MUSIL, M.; MUSIL, P.
- HDR merger SW; Software pro skládání a zpracování HDR snímků; software; MUSIL, M.; NOSKO, S.; MUSIL, P.
- Bilateral filter IP core; Bilaterální filtr pro HDR tone mapping (IP core); software; NOSKO, S.; MUSIL, M.; MUSIL, P.
- WaldBoost Detector; HW detektor objektů ve videu; software; MUSIL, M.; MUSIL, P.; JURÁNEK, R.; ZEMČÍK, P.

Stays Abroad

- 2013 COST IC1005 - HDR Summer school, Technological Educational Institute of Crete, Heracleion, Greece, one week
- 2014 University of Warwick, Coventry, United Kingdom, 2 stays (each one week)
- 2019 Betl and Road Summer School on Technological Innovation, Beijing Institute of Technology, Beijing, China, two week

Bibliography

[1] J. Cho, S. Mirzaei, J. Oberg, and R. Kastner, "Fpga-based face detection system using haar classifiers," in *FPGA*, 2009.

[2] J. Granát, A. Herout, M. Hradiš, and P. Zemčík, "Hardware acceleration of adaboost classifier," in *Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, 2007, pp. 1–12.

[3] M. Hiromoto, H. Sugano, and R. Miyamoto, "Partially parallel architecture for adaboost-based detection with haar-like features," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, 2008.

[4] C. Kyrkou and T. Theocharides, "A flexible parallel hardware architecture for adaboost-based real-time object detection," in *VLSI Systems*, 2011.

[5] C. Huang and F. Vahid, "Scalable object detection accelerators on fpgas using custom design space exploration," *SASP*, 2011.

[6] B. Brousseau and J. Rose, "An energy-efficient, fast fpga hardware architecture for opencv-compatible object detection," in *Field-Programmable Technology (FPT)*, 2012.

[7] S. Jin, D. Kim, T. T. Nguyen, D. Kim, M. Kim, and J. W. Jeon, "Design and implementation of a pipelined datapath

for high-speed face detection using fpga," *IEEE Transactions on Industrial Informatics*, vol. 8, pp. 158 – 167, 2012.

[8] P. Zemčík and M. Žádník, "Adaboost engine," in *FPL*, 2007.

[9] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *CVPR*, 2001.

[10] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," in *CVPR*, 2005.

[11] J. Šochman and J. Matas, "Waldboost - learning for time constrained sequential detection," in *CVPR*, 2005.

[12] E. Ohn-Bar and M. M. Trivedi, "To boost or not to boost? on the limits of boosted trees for object detection," *CoRR*, vol. abs/1701.01692, 2017.

[13] F. Kadlček and O. Fučík, "Automatic synthesis of small adaboost classifier in fpga," in *Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2013.

[14] H.-C. Lai, M. Savvides, and T. Chen, "Proposed fpga hardware architecture for high frame rate face detection using feature cascade classifiers," in *BTAS*, 2007.

[15] P. Musil, R. Juránek, M. Musil, and P. Zemčík, "Cascaded stripe memory engines for multi-scale object detection in fpga," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 1, pp. 267–280, 2019.

[16] D. Barina, P. Musil, M. Musil, and P. Zemcik, "Single-loop approach to 2-d wavelet lifting with jpeg 2000 compatibility," in *2015 International Symposium on Computer Architecture and High Performance Computing Workshop (SBAC-PADW)*, 2015, pp. 31–36.

[17] P. Zemčík, P. Musil, and M. Musil, *High Dynamic Range Video*. Elsevier Science, 2016, pp. 145–154.

[18] S. Nosko, M. Musil, P. Musil, and P. Zemčík, "True hdr camera with bilateral filter based tone mapping," in *SCCG '17: Spring Conference on Computer Graphics 2017*. Association for Computing Machinery, 2017, pp. 1–9.

[19] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[20] P. Zemcik, R. Juranek, P. Musil, M. Musil, and M. Hradis, "High performance architecture for object detection in streamed videos," in *Field Programmable Logic and Applications (FPL)*, 2013.

[21] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.

[22] R. Juránek, A. Herout, and P. Zemčík, "Impelementing local binary patterns with simd instructions of cpu," in *Proceedings of Winter Seminar on Computer Graphics*. West Bohemian University, 2010, p. 5.

[23] A. Herout, R. Jošth, R. Juránek, J. Havel, M. Hradiš, and P. Zemčík, "Real-time object detection on cuda," *Journal of Real-Time Image Processing*, vol. 2010, no. 1, pp. 1–12, 2010.

[24] C. Kyrkou, C. Ttofis, and T. Theocharides, "Fpga-accelerated object detection using edge information," *International Conference on Field Programmable Logic and Applications (FPL 2011)*, pp. 167 – 170, September 2011.

[25] S. Martelli, D. Tosato, M. Cristani, and V. Murino, "Fast fpga-based architecture for pedestrian detection based on covariance matrices," in *Image Processing (ICIP)*, 2011.

[26] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, p. 1532–1545, Aug. 2014.

[27] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[28] C. Kyrkou, C.-S. Bouganis, T. Theocharides, and M. M. Polycarpou, "Embedded hardware-efficient real-time classification with cascade support vector machines," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, pp. 99–112, 2016.

[29] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "Fpga implementation of real-time head-shoulder detection using local binary patterns, svm and foreground object detection," in *Design and Architectures for Signal and Image Processing (DASIP)*, 2012.

[30] C. Yeh, C. Lin, K. Muchtar, H. Lai, and M. Sun, "Three-pronged compensation and hysteresis thresholding for moving object detection in real-time video surveillance," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 4945–4955, 2017.

[31] D. Bouris, A. Nikitakis, and I. Papaefstathiou, "Fast and efficient fpga-based feature detection employing the surf algorithm," in *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, 2010, pp. 3–10.

[32] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *ISCAS*, 2010.

[33] Y. Sato and Y. Kuriya, "Multi-scale elastic graph matching for face detection," *Journal on Advances in Signal Processing*, vol. 2013, p. 175, 11 2013.

[34] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *Proceedings of the British Machine*

Vision Conference. BMVA Press, 2009, pp. 91.1–91.11, doi:10.5244/C.23.91.

[35] Y. Yazywa, T. Yashimi, T. Tsuzuki, T. Dohy, Y. Yamauchi, T. Yamashita, and H. Fujiyoshi, "Fpga hardware with target-reconfigurable object detector," *IEEE Transactions on Information and Systems(IEICE)*, vol. 98, no. 9, pp. 1637–1645, 2015.

[36] X. Ma, W. A. Najjar, and A. K. Roy-Chowdhury, "Evaluation and acceleration of high-throughput fixed-point object detection on fpgas," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 1051–1062, 2015.

[37] Y. Said and M. Atri, "Efficient and high-performance pedestrian detector implementation for intelligent vehicles," *Intelligent Transport Systems(IET)*, vol. 10, pp. 438–444, 2016.

[38] Z. Xu, R. Shi, Z. Sun, Y. Li, Y. Zhao, and C. Wu, "A heterogeneous system for real-time detection with adaboost," in *High Performance Computing and Communications*, 2016.

[39] M. Bilal, A. Khan, M. U. Karim Khan, and C. Kyung, "A low-complexity pedestrian detection framework for smart video surveillance systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 10, pp. 2260–2273, 2017.

[40] J. Yang, Y. Yang, Z. Chen, L. Liu, J. Liu, and N. Wu, "A heterogeneous parallel processor for high-speed vision chip," *IEEE TSCVT*, 2017.

[41] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[42] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016.

[43] S. Bhattarai, A. Madanayake, R. J. Cintra, S. Duffner, and C. Garcia, "Digital architecture for real-time cnn-based face detection for video processing," in *CCAA*, June 2017, pp. 1–6.

[44] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded fpga platform for convolutional neural network," in *FPGA*, ser. FPGA '16. New York, NY, USA: ACM, 2016.

[45] C. Zhang, D. Wu, J. Sun, G. Sun, G. Luo, and J. Cong, "Energy-efficient cnn implementation on a deeply pipelined fpga cluster," in *ISLPED*. New York, NY, USA: ACM, 2016, pp. 326–331.

[46] M. Hradiš, A. Herout, and P. Zemčík, "Local rank patterns - novel features for rapid object detection," in *Proceedings of International Conference on Computer Vision and Graphics 2008*, ser. Lecture Notes in Computer Science, 2008, pp. 1–2.

[47] R. Juránek, M. Hradiš, and P. Zemčík, *Real-Time Systems*. InTech Education and Publishing, 2012, ch. Real-Time Object Detection with Classifiers, p. 21.

[48] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *Int. J. Comput. Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[49] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Gray scale and rotation invariant texture classification with local binary patterns," in *ECCV '00: Proceedings of the 6th European*

Conference on Computer Vision-Part I. London, UK: Springer-Verlag, 2000, pp. 404–420.

[50] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, no. 3, pp. 297–336, 1999.

[51] K. Turkowski, "Properties od surface-normal transformations," in *Graphics Gems*, 1990.

[52] J. Sochman and J. Matas, "Learning fast emulators of binary decision processes," *International Journal of Computer Vision*, vol. 83, no. 2, pp. 149–163, June 2009.

[53] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas, "A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera," in *ITS Conference*, Sep. 2012, pp. 975–982.

[54] A. Broggi, E. Cardarelli, S. Cattani, P. Medici, and M. Sabbatelli, "Vehicle detection for autonomous parking using a soft-cascade adaboost classifier," in *2014 IEEE IVSP*, June 2014.