HIGH PERFORMANCE ARCHITECTURE FOR OBJECT DETECTION IN STREAMED VIDEOS

Pavel Zemčík, Roman Juránek, Petr Musil, Martin Musil, Michal Hradiš

Department of Computer Graphics and Multimedia, Faculty of Information Technology,
Brno University of Technology, Brno, CZ
{*zemcik,ijuranek,imusilpetr,imusil,ihradis*}*@fit.vutbr.cz*

ABSTRACT

In this paper, we introduce a novel 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 processing of streamed data and low resource consumption. We implemented the engine in FPGA and we show that it can process 640×480 pixel video streams at over 160 fps without the need of external memory. We evaluate the design on the face detection task, compare it to state of the art designs, and discuss its features and limitations.

1. INTRODUCTION

One of most widely used methods for object detection in videos [1] uses a classifier to evaluate every suitable subwindow of an input image in order to determine whether the area contains a target object or not. In case of objects occurring in multiple sizes, the detection must be performed in multiple scales Several authors have proposed hardware implementations of object detectors. In most cases, they use cascade of boosted classifiers [2, 3] proposed by Viola [1], but other approaches to the detection, such as neural networks [4], are also used. Kim et al. [3] introduced a real-time eye detector for FPGA based on an AdaBoost classifier and MSC local image features. Cho et al. [5] proposed architecture implementing AdaBoost cascade detector with Haar features. In their approach, large memory is used to perform multi-scale detection on a pyramid of integral images. Huang [6] also used a cascade classifier with Haar features and they tried to reduce resource requirements of integral image memory by a multiplex network and by constraints during the training. A low resources detection system was proposed by Zemčík [7]. They used WaldBoost classifier, Local Rank Differences (LRD) [8] image features, they saved memory by storing a small image strip. Multi-scale detection was made possible by an external DSP unit which precalculated an image pyramid. Kyrkou [2] introduced an AdaBoost cascade detector which combines two approaches to perform multi-scale detection – image downscaling and scaling of a scanning window. Such a design, however, does have relatively high consumption of resources and memory as it uses classifiers with Haar features and normalization must be performed.

In our design, we improve multi-scale detection, resolution of input image, and stream processing compared to [7]. We use very simple feature extractors – Local Binary Patterns (LBP) [9] and Local Rank Differences (LRD) [8]. We replaced the cascade of boosted classifiers with a WaldBoost [10] classifier, which provides improved detection speed and accuracy. We store only a narrow stripe of image which fits in the memory on chip. The architecture can handle streamed input, process it, and add detection results into the video stream. The engine can process 640×480 pixel video stream faster than in real-time while consuming very little power. It consumes only a modest amount of FPGA resources and thus multiple instances of the detection engine can be implemented in a single FPGA in order to boost the performance, or to enable detection of multiple object types. Special focus was put on minimization of the expensive memory structures and on low energy demand. The detection rate measured on the task of face detection is comparable to other similar architectures.

2. OBJECT DETECTION WITH CLASSIFIERS

The best known and most widely used algorithm for object detection with classifiers is *cascade* of boosted classifiers proposed by Viola [1]. The cascade subdivides the classifier into several increasingly complex classifiers (called *stages*). After the evaluation of a stage, a decision about the class of the input image is made. Early stages can reject the majority of background samples and thus the computational complexity is kept quite low. Every stage is composed from very simple elementary classifiers called *weak hypotheses*. A more advanced method, *Soft cascade* [10], learns a *sequential classifier* which makes decisions about an image class after evaluating every weak hypothesis. In this work, we use WaldBoost algorithm [10], which produces a soft cascade classifier represented by a sequence of weak hypotheses $h^{(t)} = (f^{(t)}, \phi^{(t)}, \alpha^{(t)}, \theta^{(t)})_{t=1}^{T}$ where T is the total

978-1-4799-0004-6/13/$31.00

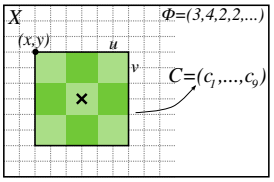


Fig. 1: Feature extraction form an image. Cells C are taken from X according to ϕ. Shades of green are used to make the individual cells visible. Central cell c is marked by a cross.

number of weak hypotheses. Every hypothesis contains a feature extraction function $f^{(t)}$, parameters of feature extraction $\phi^{(t)}$, a list of responses $\alpha^{(t)}$ and a threshold $\theta^{(t)}$. The response of t-th weak hypothesis $h^{(t)}$ on image sub-window X is obtained by indexing $\alpha^{(t)}$ using the response of the corresponding image feature $f^{(t)}(X,\phi^{(t)})$.

$$H^{(t)}(X) = \sum_{k=1}^{t} \alpha^{(k)}_{f^{(k)}(X,\phi^{(k)})} \quad (1)$$

$$S^{(t)}(X) = \begin{cases} 0 & H^{(t)}(X) < \theta^{(t)} \\ S^{(t+1)}(X) & \text{otherwise} \end{cases} \quad (2)$$

The response of the strong classifier $H^{(t)}$ in every step is accumulated (1) and compared to $\theta^{(t)}$ (2). If the sum falls below the threshold, the evaluation ends as the input is likely to be the background. If the evaluation reaches the last weak hypothesis, the final decision is positive and it is likely that the object was detected. Given an input image, the classifier analyzes its sub-images X from every position. Every image area is classified and the positive final decisions are treated as detected objects. Multi-scale detection is usually solved either by *scaling of classifier window* or *scaling of the image*. In this work, we explored both options.

So far most widely used image features are Haar features [1] as they proved to be a good information extraction tool for various tasks. In this work, we use LBP [9] and LRD [8] features, which perform comparably to Haar features, and they offer interesting properties from the implementation point of view [8, 7]. The features are parametrized by their geometrical properties. LBP features are defined as $\phi^{LBP} = (x,y,u,v)$ where x,y corresponds to the position in image X, and $u,v \in \{1,2\}$ is the size of feature cells in pixels. LRD features additionally contain identifiers of two selected cells a and b, thus $\phi^{LRD} = (x,y,u,v,a,b)$. A feature is evaluated from 3×3 grid of cells C represented by sums of their values in X (see Figure 1). X and ϕ impose C as mentioned above. LBP features (3) compare the central cell c with all the other cells C^B. The results are concatenated to produce an 8-bit word. LRD features (4) calculate the ranks of the cells a and b, and subtract them, producing results in the range $\langle -8;8 \rangle$. Rank is the number of positive comparisons of the cell value to all other cells. Term $[\cdot]$ returns 1 when the comparison is true, 0 otherwise.

In both cases, the results of feature evaluation do not significantly depend on brightness and contrast in the image. Therefore, normalization of the image is not required. All the operations used in the feature evaluation are very simple (comparison, addition, subtraction) and thus, the hardware implementation is straightforward, and it consumes very few resources as shown in experiments.

$$f^{LBP}(X,\phi^{LBP}) = \sum_{i=1}^{8} 2^{i-1} \left[C_i^B > c\right] \quad (3)$$

$$f^{LRD}(X,\phi^{LRD}) = \sum_{i=1}^{9} [C_i > C_a] - \sum_{i=1}^{9} [C_i > C_b] \quad (4)$$

3. PROPOSED ARCHITECTURE

The engine, shown in Figure 2a, works as a programmable automata driven by an 64 bit instruction set. It evaluates given classifier on every image position, (see equations (1) and (2)). A position is marked as positive only if all weak hypotheses are evaluated. The locations with the positive response can be sent out in the form of a bitmap or a list of coordinates. The instruction code for a stage t stores parameters for feature extraction $\phi^{(t)}$, stage identifier t for addressing table $\alpha^{(t)}$ and $\theta^{(t)}$, and additional information for engine control. The engine is pipelined to parallelize the evaluation of weak hypotheses.

3.1. Memory Access

The engine stores a narrow image stripe with a small part of the processed image and its scaled versions (see Figure 3c). The memory is organized as a circular buffer of rows. Each 36-bit memory cell stores 6 image pixels (6 bits per pixel). This conversion can be reflected in the training and does not significantly influence the detection rate. Memory addressing is optimized for the reading of 6×6 pixel blocks in every clock cycle. In our experimental design, Xilinx Spartan 6 series FPGA is used, and the memory is implemented in 12 BRAMs that store 24 or 32 lines of the image.

3.2. Image Scaling

We explored two approaches for multi-scale object detection that can be implemented in the engine.

Fine Image Scaling creates the first octave of the pyramid using a fine scaling unit (we use scale factor of 5/6). The subsequent octaves are calculated from the previous ones by downscaling with a factor of 1/2. This is illustrated in Figure 3a. A drawback of this approach is still relatively large memory requirement for a complete image pyramid, and consumption of resources for the scaling units.

Coarse Image Scaling scales image in memory by the factor of 1/2, as illustrated in Figure 3b. Fine scaling is emulated by using classifiers with different window sizes. To cover the same scales as in the case of a 5/6 scaling unit, four classifiers must be used. The advantages of this strategy include a decrease of memory space required for the

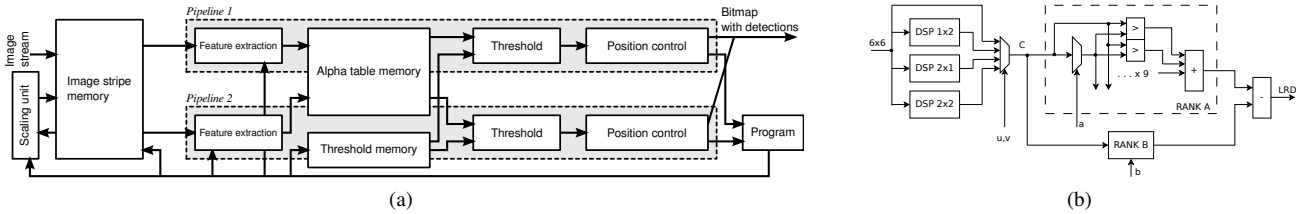


Fig. 2: (a) Block diagram. The data for *Feature Extraction* blocks are loaded from *Stripe Image Memory*, hypotheses are evaluated using *Alpha Table Memory*, *Threshold* compares classifier responses to thresholds. *Instruction Memory* holds the program; (b) circuit for LRD feature evaluation implementing (4).

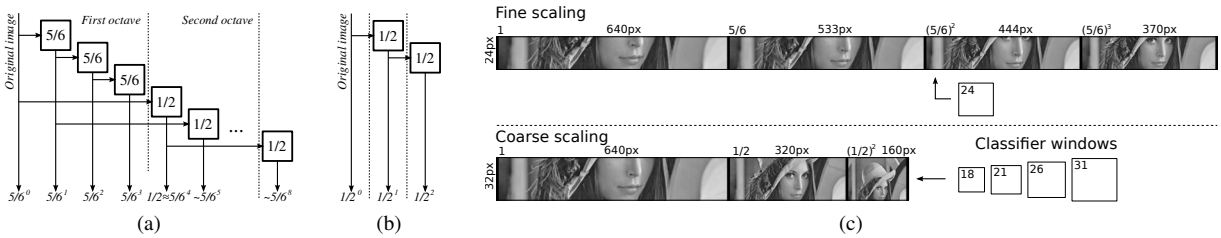


Fig. 3: Illustration of image scaling approaches. (a) Fine scaling utilizing 5/6 and 1/2 scaling units; (b) coarse scaling with 1/2 scaling units; and (c) content of the image buffer stripe in the memory for fine scaling (top), and coarse scaling (bottom).

images and the reduced resource consumption as fine scaling units are no longer required. The disadvantages, on the other hand, include the need to store more classifier definitions and to evaluate the classifiers at more positions.

In the experiments in Section 4, we refer to different versions of the engine as 5/6 or 1/2.

3.3. Feature Extraction

The detection engine implements LBP and LRD image features described in Section 2. The size of the feature cell (u, v in $\phi^{(t)}$) is limited to a max. of 2 pixels and thus the features are limited to a 6×6 pixel area. Figure 4 shows four versions of cells that can be extracted from a feature area.

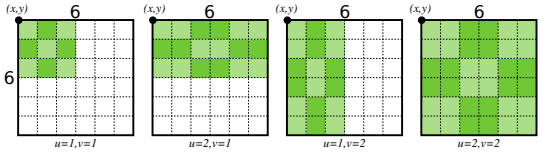


Fig. 4: Four configurations of cells C extracted from a 6×6 pixel area on position x, y.

The block scheme of LRD feature evaluation is shown in Figure 2b. DSP blocks extracts the four versions of cells C. One version is selected for evaluation according to u, v. The ranks of cells a and b are calculated as the number of positive comparisons of the cell with all other cells. The ranks are subtracted to obtain the feature response. Evaluation of an LBP feature is similar – parallel comparison of the central cell with the cells at the boundary. The response of a weak hypothesis is obtained from the look-up table $\alpha^{(t)}$.

4. EXPERIMENTS AND RESULTS

The detection architecture was experimentally synthesized in a Xilinx Spartan 6 LX45T FPGA and evaluated on the Xilinx SP605 board[1]. Ethernet camera CAMEA Modicam M621 provides a 640×480 pixel video input at 60 fps. The performance of the engine was evaluated on a face detection task. The detectors were composed from 128 weak hypotheses. Values in $\alpha^{(t)}$ and $\theta^{(t)}$ were quantized during the training. The engine contains two multi-stage pipelines for weak hypothesis evaluation Image data, instruction memory and classifier definitions are shared between the two pipelines while the other units are replicated. Different versions of the engine were used to demonstrate how the scaling strategy and image features affect the performance and resource consumption. The combinations were: LBP 5/6, LRD 5/6, LBP 1/2, and LRD 1/2. Each combination refers to a feature type and scaling unit as described in Section 3.2.

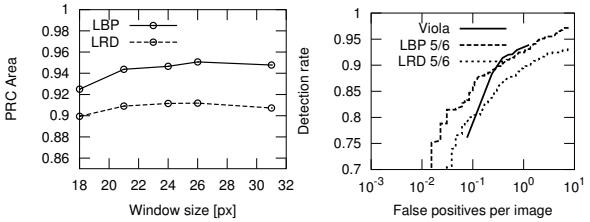
4.1. Evaluation of Classifiers Properties

In the versions with fine scaling, we use detectors with a 24-pixel window. In versions with coarse scaling, four detectors are used to emulate the 5/6 scaling step (18, 21, 26 and 31-pixel windows). Figure 5a summarizes detection performance of all the detectors. Figure 5b shows that the detection rate of our architecture is comparable to state of the art architectures which use detector by Viola [1]. Our detectors are composed from only 128 weak hypotheses, whereas detector by Viola contains 6,061 weak hypotheses.

[1] VHDL sources and detectors can be downloaded form `http://medusa.fit.vutbr.cz/fpga-engine`

	FPS	Features	Scaling method	Scale factor	Freq. [MHz]	BRAMs	LUTs	Regs.	FPGA
Huang [6]	—	Haar	Img. scaling	1.2	65	—	80000	—	Virtex5 LX155T
Cho [5]	7	Haar	Img. scaling	1.2	—	41	66900	21900	Virtex5 LX110T
Kim [3]	50	MCT	Img. scaling	—	106	18	133000	45700	Virtex5 LX330T
Kyrkou [2]	40	Haar	Img./feature scaling	1.33	100	24	25800	23800	Virtex2 XC2VP30
Lai [11]	143	Haar	Img. scaling	1.25	126	44	20900	7800	Virtex2 XC2VP30
Zemcik [7]	22	LRD	None	None	—	14	2980	—	Virtex2 250
Our LRD 5/6	164	LRD	Img. scaling	1.2	152	31	7373	1732	Spartan6 LX45T
Our LRD 1/2	107	LRD	Img. scaling	1.2	163	29	7014	1673	Spartan6 LX45T

Table 1: Comparison of our engine with similar designs. The FPS column shows performance on a 640×480 pixel image.



(a) (b)

Fig. 5: Evaluation on MIT+CMU face dataset. (a) Accuracy of our detectors as a function of detector size; (b) ROC curves of our detectors compared to cascade detector [1].

	FPGA Resources			Performance	F
	Registers	LUTs	BRAMs	[MHz]	[fps]
LBP 1/2	1678 (3%)	7098 (26%)	77 (66%)	163	91
LBP 5/6	1737 (3%)	7405 (27%)	43 (37%)	152	131
LRD 1/2	1673 (3%)	7014 (26%)	29 (25%)	163	107
LRD 5/6	1732 (3%)	7373 (27%)	31 (27%)	152	164

Table 2: Device utilization summary on Spartan6 LX45T.

4.2. Resource Consumption

Table 1 shows the comparison of our architecture to similar architectures. Two variants of the presented design are listed – the highest performance variant (LRD 5/6) and the least resources (LRD 1/2). Compared to [5, 6, 3] our engine consumes only a small fraction of resources. Our engine requires more BRAM blocks than the earlier engines, but it does not need any external blocks Unlike the earlier engines, our engine implements stream processing in a single chip.

Table 2 summarizes the resource consumption of the four configurations of the engine and estimation of maximal throughput F. It shows that LBP versions use more BRAM blocks because $\alpha^{(t)}$ in LBP weak hypothesis contain 256 items and only 32 in LRD. This can be observed especially in the versions with coarse scaling versions, which need more memory to store the four classifiers. The difference between the versions using the LRD features are marginal. Less demanding design versions are those with LRD classifiers, especially the LRD 1/2. The highest performance to resource consumption ratio is provided by the LRD 5/6 variant as its performance reaches over 160 fps. According to simulation in XPower, power consumption of each variant is 0.5 W.

5. CONCLUSION

This paper presented a novel architecture for real-time object detection in streamed video using a scanning window detector. The main achievements of the new architecture include very high performance, multi-scale detection, extremely compact design, and low consumption of hardware resources. Such a design is possible thanks to the novel approach to feature extraction. Multi-scale detection is achieved either by fine scaling of the image and use of a fixed size classifier, or by coarse scaling and the use of four detectors. The LRD 5/6 version is capable of processing 640 × 480 pixel video stream at over 160 fps. The design is suitable for applications that can benefit from embedded design and small power consumption. Future research includes improvements in the multi-resolution design, further reduction of resource consumption, implementation of multiple object classifiers in one engine, and algorithmic improvements.

6. ACKNOWLEDGMENTS

This work has been supported by projects "ConstRaint and Application driven Framework for Tailoring Embedded Real-time Systems", Artemis JU, 7H12006, "Tools and Methods for Video and Image Processing for the Fight against Terrorism", VG20102015006, "Visual Computing Competence Center", TE01010415, and CAMEA spol. s.r.o.

7. REFERENCES

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

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

[3] D.-K. Kim, J.-H. Jung, T. T. Nguyen, D.-J. Kim, M.-S. Kim, K.-H. Kwon, and J.-W. Jeon, "An fpga-based parallel hardware architecture for real-time eye detection," *JSTS*, 2012.

[4] K. Curran, X. Li, and N. M. Caughley, "The use of neural networks in real-time face detection," *IJCS*, 2005.

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

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

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

[8] A. Herout, P. Zemčík, M. Hradiš, R. Juránek, J. Havel, R. Jošth, and M. Žádník, "Low-level image features for real-time object detection," *In-Tech*, 2010.

[9] L. Zhang, R. Chu, S. Xiang, S. Liao, and S. Z. Li, "Face detection based on multi-block lbp representation," in *ICB*, 2007.

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

[11] 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.