Bio-inspired heterogeneous architecture for real-time pedestrian detection applications

Luca Maggiani, Cédric Bourrasset, Jean-Charles Quinton, François Berry, Jocelyn Sérot

Draft manuscript No.
(will be inserted by the editor)

Luca Maggiani [1,2], Cédric Bourrasset [2], Jean-Charles Quinton [2,3], François Berry [2], Jocelyn Sérot [2]

Bio-inspired Heterogeneous Architecture for Real-Time Pedestrian Detection Applications

the date of receipt and acceptance should be inserted later

Abstract Along with the development of powerful processing platforms, heterogeneous architectures are nowadays permitting new design space explorations. In this paper we propose a novel heterogeneous architecture for reliable pedestrian detection applications. It deploys an efficient Histogram of Oriented Gradient pipeline tightly coupled with a neuro-inspired spatio-temporal filter. By relying on hardware-software co-design principles, our architecture is capable of processing video sequences from real-word dynamic environments in real-time. The paper presents the implemented algorithm and details the proposed architecture for executing it, exposing in particular the partitioning decisions made to meet the required performance. A prototype implementation is described and the results obtained are discussed with respect to other state of the art solutions.

1 Introduction

As reported by the World Health Organization, in 2013 about 1.2 million people died in accidents [34], most of them attributable to human errors or drivers distractions. Although the big efforts spent since 1990s to develop safety technologies solutions such as airbags, antilock brakes, traction control, etc, human casualties in the road environment are still too high. In the latest years, a new level of on-board intelligence, named Advanced Driver Assistant System (ADAS), has been developed to actively assist the driver in order to improve the road security. The main idea beneath ADAS technologies is to extend the human visual information with one or more cameras mounted outside the cabin. These cameras are autonomously processing the images to reveal possible threats and eventually prevent the accident by communicating with the on-board ADAS system. Along with the evolution of computer vision capabilities, vision sensors are becoming more and more important to extend the ADAS environmental understanding. In [6, 20, 36] detailed surveys of the available detection techniques have been presented. In particular, two main challenges arise from the application point of view. First, vision-enabled ADAS systems should provide *reliable* detection information. This aspect obviously is the essential condition to rely on computer vision deployment and already represents a challenging task so far. But, because they operate in an environment which intrinsically changes very rapidly, they should also provide this information in real-time. Most of the time, given the complexity of the involved computer vision algorithms, this requires a computing power which is still not available from commercial-of-the-shelf (COTS) embedded processing platforms.

In developing effective ADAS systems – and in many other computer vision applications as well – three main processing stages can be identified as shown in Fig. 1. The sparse and noisy image information is firstly processed to extract significant features. Those features are then analysed and classified into known patterns to recover higher level image characteristics. Finally, a post processing phase includes the complete scene comprehension and understanding, towards a virtual events perception.

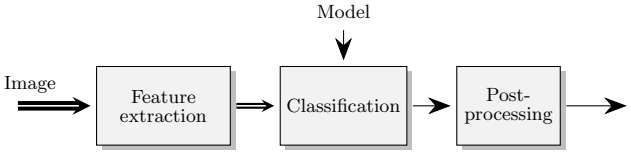


Fig. 1: Pedestrian detection overview.

Feature extraction Feature extraction is generally based on the computation of so-called *descriptors* and several

[1] TeCIP Institute, Scuola Superiore Sant'Anna, Pisa, Italy
[2] Institute Pascal, Universite Blaise Pascal, Clermont Ferrand, France
[3] Laboratory Jean Kuntzmann, Grenoble Alpes University, France

kinds of descriptors have already been proposed to increase detection reliability. In the context of pedestrian detection, descriptors relying on spatial orientation patterns, such as Local Binary Patterns (LBPs) [28] and Histograms of Oriented Gradient (HOG) [5] have been specially studied. With LBPs, neighbor pixel comparisons are encoded with binary strings to describe the local pattern distribution. Since the result is based on relative luminance, this method limits the dependence to global luminance variation. HOG-based methods, on the other hand, aim at improving the descriptor robustness for structured image patterns. For this, the local spatial characteristics are described with a distribution of local intensity gradients within uniformly spaced cells. The resulting gradient histogram within each cell, possibly normalised, represents an element of the final HOG descriptor. Due to its invariance to luminance variation and good detection performance [3], the HOG descriptor is well suited to mobile vision applications. In this paper, an optimized version of the HOG pipeline is deployed to reduce the computational requirements for real-time embedded ADAS deployments.

Classification Feature extraction is generally followed by a classification step. Classification aims at evaluating whether specific patterns, whose shape is supposed to be captured by the descriptors, occur in the input image. Machine-learning based systems are generally used to find correlations between the computed descriptors and a reference model. These systems typically emit detection results with their associated confidence interval. The higher is the confidence in the result, the higher is the probability of a correct classification. In the context of pedestrian detection, Support Vector Machines (SVM) are commonly used for performing the classification step, as a widespread and efficient supervised learning technique, based on strong mathematical foundations. The goal of this technique is to separate sets of descriptors in two classes. Once the reference model has been generated (with an off-line training phase), SVM produces detection classifications by labelling the input feature descriptors. Although algorithmically simple, application of SVM to image classification raises challenging implementations issues, especially if real-time performance level is required, as in pedestrian detection applications.

Post-processing As previously introduced, the classification reliability is one of the most critical point for pedestrian detection system. Classically, reliability is assessed by means of two indicators : the true positive detection rate (TPR) and a false positive detection rate (FPR). The former corresponds to the "sensitivity" of the detection process and the latter to its "specificity". Best methods are supposed to achieve the highest TPR while keeping the lowest FPR.

Reliability numbers obtained with pure SVM-based algorithms are generally not sufficient for an effective ADAS deployment. In response, several post-processing techniques have been proposed to improve this reliability, essentially by reducing the FPR.

The contribution of this paper is threefold. First, we describe a reformulation of the HOG computation process allowing a very efficient implementation on deeply pipelined parallel architectures. Second, we introduce an original method based on probabilistic filtering method, exploiting spatio-temporal coherence between consecutive image detections, for significantly improving the reliability of the SVM-based classification process. Third, we describe an heterogeneous architecture implementing these ideas and providing both real-time processing performance and reliable detection results for pedestrian detection applications. The term "bio-inspired" comes from the fact that this hybrid architecture, which relies on the transformation of a large amount of unstructured visual data into smaller, more abstract and robuts representations of objects of interest, somehow mimicks the transition from homogeneous retinal information to abstract and sparse coding in higher cortices of the brain (transition which has been extensively studied in the field of neuro-sciences).

The rest of the paper is organized as follows. In section 2 several state-of-the-art pedestrian detection solutions are presented. Our proposed solution is compared with previous works to show the benefits of a bio-inspired heterogeneous architecture. The section 3 introduces the mathematical background of the chosen algorithms and the contribution of the paper to the state-of-the-art. Our implementation is described in section 4. Section 5 gives and discusses preliminary results in terms of detection quality, hardware resource usage and performance, obtained with a prototype implementation running on a platform embedding a Altera Cyclone 5 FPGA and a ARM Cortex A9 CPU. Section 6 concludes the paper.

2 Related Work

Because of their potentially pervasive applications, pedestrian detection systems are a very active research area. Although SVM-based system have already shown good detection performance with real-world images [16, 6], work is still on-going in order to improve the detection reliability. For this, two main directions have been suggested. The first is to extract and exploit several sets of feature descriptors. The second is to improve the robustness of the extracted feature descriptors.

Work described in [38, 37, 31, 14] belong to the first category. In particular, in [38] and [37] HOG-LBP and HOG-Local Self-Similarity (LSS) algorithms are respectively deployed to increase the detection accuracy with multiple algorithm predictions. A further improvement has been recently proposed in [31], where a HOG-LBP

detection process is applied simultaneously and separately on the three channels of color images. These implementations rely on weighted inferences among the available detections to improve the final decision result. It can be noted, however, that requiring simultaneous computation of several feature sets places severe constraints on the implementation, especially if real-time behavior is targeted. In [14], the computation of a combined HOG-Discrete wavelet transform (DWT) is restricted to selected ROIs in image in order to reduce the global computation latency. A similar approach is presented in [22], where ROI-based pedestrian multi-view pose evaluations are processed with a HOG-LBP pipeline followed by a non linear SVM. Nonetheless, ROI-based approaches inherently reduce the "active" image part. In an environment which changes very rapidly, finding an acceptable trade-off between global computation time and the number of active ROIs is a complex problem, especially for time critical applications.

The second investigated approach for improving detection reliability concerns the descriptor robustness. Basically, the idea is to extend the feature descriptor dimension in order to improve its robustness in real-world deployment. In [35], for instance, the HOG descriptor is improved by taking into account similarity on the local representation of contours. Since the pedestrian appearance usually presents symetry properties, a geometric feature description is added. A two-level cascade of SVM classifiers is then applied to the extended descriptor. Even though this method improves the performance with respect to a classic HOG descriptor, the processing performance overhead once again reduces the applicability of the method in embedded platforms. Another approach is to augment HOG descriptors with temporal informations obtained from video sequences. In [15], for example, the authors aggregate several descriptors obtained by different techniques to extract temporal information from images. The 3DHOG descriptor proposed in [21] for characterizing motion features with a co-occurence spatio-temporal vector also belongs to this category. The HOGHOF descriptor using histogram of optical flow and the STHOG (Spatio-temporal histogram of oriented gradient) proposed by the author both increase the discriminative nature of the usual HOG descriptor space, at the price, here again, of a highest computation cost.

To address performance issues, FPGA and GPU implementations have been proposed. For instance, in [24] a hybrid CPU/GPU architecture is described, achieving a speedup factor of 30x with respect to a conventional CPU implementation. In [2] another HOG GPU optimized implementation is proposed to tackle both the processing performance and reliability issues. However, given their energetic requirements, GPU-based solutions are more suited to high-end, non-embedded applications. Solutions based on reconfigurable logic, FPGAs in particular, generally show a better performance/Watt ratio, making them more attractive for embedded applications. In [11], for example, a FPGA-based HOG-SVM pedestrian detection system is proposed. However, in this realization, the achieved detection performance does not meet the reliability requirements for a real-world pedestrian detection scenario, due to the approximations resulting from the fixed-point computations that are usually implemented within hardware pipelines. The impact of arithmetic approximations has been further investigated in [23], where several fixed-point implementations have been evaluated to better explore the trade-off between resource usage and accuracy. The conclusion is that, although current FPGA implementations achieve the highest processing performance, they do no cope with CPU-based implementations in terms of detection reliability.

In response to the aforementioned issues, so-called *heterogeneous* platforms, combining general purpose processors and hardware accelerators, have been proposed. The idea is to exploit highly parallel architectures for low-level pixel-wise operations and general purpose processors for higher level inferences. Design methodologies for this kind of platforms have been presented in [12, 13, 30, 17], for example. Typically, design proceeds by moving computationally intensive parts of the application from software to hardware accelerators, while keeping in software the parts having the lowest computational cost and/or the more demanding requirement in terms of arithmetic precision. In [19, 1, 4] several methods, aiming at finding the "best" partition between the hardware and software parts, have been formally analysed to distinguish their benefits in terms of development time, efficiency and architecture optimisation. The approach we propose and describe in Sec. 4 follows this so-called *co-design* approach. Comparative results (with the aforementioned work) will be given in Sec. 5.

3 Detection algorithm

As stated in Sec. 1, the detection algorithm consists in a feature extraction step followed by a classification step. The extracted features are HOG (Histogram of Oriented Gradients), computed on a dense grid of uniformly spaced cells and normalized. Practically, the input image is divided into small connected regions, called *cells*, and within each cell an histogram of gradient directions is computed. The resulting descriptors are sent to a pretrained SVM-based classification machine.

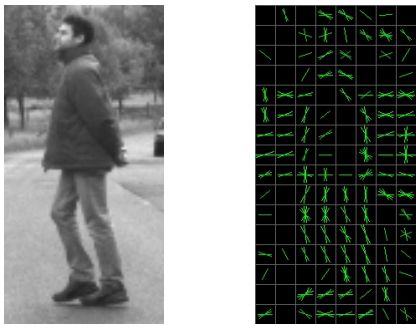
Following Dalal's formulation in [5], the 1-D spatial gradient components are first computed as follows:

$$G_x = \frac{\partial I}{\partial x} = I\big(x+1, y\big) - I\big(x-1, y\big) \qquad (1)$$
$$G_y = \frac{\partial I}{\partial y} = I\big(x, y+1\big) - I\big(x, y-1\big)$$

Then the gradient magnitude $\|\nabla I(x,y)\|$ and orientation angle θ are evaluated as in Eq. 2.

$$\|\nabla I(x,y)\| = \sqrt{G_x^2 + G_y^2} \qquad (2)$$
$$\theta = \arctan \frac{G_y}{G_x}$$

Within each cell, gradient orientations are then discretized and accumulated in a small number of histogram bins. Each histogram is finally normalized using a normalization factor derived from the mean intensity computed across a larger region of the image (block). This normalization results in better invariance to changes in illumination or shadowing [9]. The final HOG descriptor of an image is obtained by aggregating the normalized histograms.



(a) (b)

Fig. 2: (a) a pedestrian image from the Daimler dataset. (b) the associated HOG Descriptor.

Fig. 2b shows an example image and its associated HOG descriptor. It is worth to note that the foreground silhouette is clearly enhanced with respect to the background details. These features are completely meaningless without a supervised learning machine.

The classification stage compares the produced HOG descriptor with a pre-trained reference model. This results in a binary decision, that would mark the current descriptor for belonging to one of two categories. The detection reliability is then expressed as a confidence interval, that is usually exploited to assess the classifier accuracy. In this work, the Support Vector Machine classifier proposed in [33] has been used. The SVM algorithm compares the input vector descriptor with a reference model, which is produced by a supervised learning phase. The training step builds the reference model by assigning a great number of labelled examples to a specific class among a set of *classes*. The resulting model is then used online to map each descriptor to one of the predefined classes (non-probabilistic binary classification).

More formally, given a set of l data elements $\mathbf{x_1}, ..., \mathbf{x_l}$ and their corresponding classes $y_1 = y(\mathbf{x_1}), ..., y_l = (\mathbf{x_l})$ where $\mathbf{x_i} \in \mathbb{R}^d$ and $y_i = \pm 1$, the classification function can be expressed as follows:

$$y(\mathbf{x}) = sgn\left(\sum_{i=1}^{N_{SV}} y_i \alpha_i K(\mathbf{x_i}, \mathbf{x}) + b\right), 0 \leq \alpha_i \leq C \qquad (3)$$

where $K(\mathbf{x_i}, \mathbf{x})$ is a kernel function, C is a regularization constant, α_i and b are parameters given by the learning phase. N_{SV} is the number of reference features, called Support Vectors (SVs). The examples $\mathbf{x_1}, ..., \mathbf{x_l}$ are HOG feature vectors evaluated from database images and y corresponds to the associated class. In the case of a binary classification with a linear kernel, the general Eq. 3 can be simplified as follows:

$$y(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x} + b \qquad (4)$$

where $\mathbf{w}$ and b are model parameters, produced by the training phase. Note that in Eq. 4 the *sgn* operator has disappeared because what is actually used is the *distance* of the considered input with respect to the decision hyperplane.

In most published work, evaluation of object presence is performed in a limited number of regions of the image, called regions of interest (ROIs). We propose, by contrast, to perform this evaluation for all possible positions of a detection window, therefore called *sliding window*, in the camera field of view. This approach has the advantage of being able to automatically detect new objects in images, whereas ROI-based approach can only focus on predefined targets and need some kind of tracking system to adjust the position of the ROIs.

The equation 3 is then generalized to take into account the fact that the response of the SVM step is now a matrix in which each element gives the detection result for a given detection window in the original image:

$$Y = \begin{bmatrix} y_{1,1} & \cdots & y_{1,M} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,M} \end{bmatrix} = \begin{bmatrix} y_{n,m} \end{bmatrix} \qquad (5)$$

where $n \in \{1, \cdots, N\}$ and $m \in \{1, \cdots, M\}$. The values N and M respectively corresponds to the number of windows in the vertical and horizontal direction.

As stated in Sec. 1, a final filtering mechanism is often added in order to improve detection reliability. Since the SVM classifier here generates a full detection map, with a degree of confidence associated to each detection point in the image, we can directly apply a Dynamic Neural Field (DNF) model on the HOG-SVM classification results. DNFs can can be viewed as a neuro-inspired and massively parallel ways of performing probabilistic recursive estimation at the image level (mesoscopic scale in the visual cortex), instead of relying for instance on

the Bayesian filtering and tracking of a few areas of interest (Kalman or particle filter). DNFs exploit the spatio-temporal coherence of the stimuli, relying on population coding (i.e. distributed representations relying on overlapping visual receptive fields) to perform robust inference. We can indeed expect SVM outputs to display coherence between subsequent frames (movement continuity) and between nearby locations on the image (SVM input being generated from overlapping groups of HOG cells). Incorporating an evolution model of the target dynamics, DNFs can even further improve detection reliability by reducing the FPR. The following paragraphs rely on the classical stationary equation for explanatory purpose, but the reader can refer to [29] for details on the predictive version using a linear model of movement.

The classical DNF equation models the dynamics of the mean potential of the neural population in cortical columns, forming 2D fields in the visual cortex. The potential at position $\mathbf{x} \in X$ and time t on such a field is defined by $u(\mathbf{x}, t)$, while the input stimulation of the field is set to $y(\mathbf{x}, t)$, corresponding to SVM outputs over the entire image for the frame at time t. A simplified version of the one-layer DNF equation gives:

$$\tau \frac{\partial u(\mathbf{x}, t)}{\partial t} = -u(\mathbf{x}, t) + c(\mathbf{x}, t) + y(\mathbf{x}, t) \tag{6}$$

where c is a lateral competition term leading to the selection/detection of targets, and defined by:

$$c(\mathbf{x}, t) = \int_{x' \in X} w(\mathbf{x}, \mathbf{x'}) \sigma(u(\mathbf{x'}, t)) d\mathbf{x'} \tag{7}$$

where σ is a non linear activation function (classically a sigmoid function), and $w(\mathbf{x}, \mathbf{x'})$ is the lateral connection weight function satisfying Eq. 8. Excitatory standard deviation a controls the expected size of the target, thus allowing for multi-scale tracking, while the inhibitory standard deviation b determines the minimal distance between acquired targets. A and B are respectively the amplitudes of the excitatory and inhibitory components of the kernel. They thus control/weight the influence of the lateral competition relatively to the noisy input data (i.e. here the outputs from the SVM). In probabilistic terms, they thus weight the prior distribution on target location with observations from the current frame.

$$w(\mathbf{x}, \mathbf{x'}) = A e^{-\frac{|\mathbf{x}-\mathbf{x'}|^2}{a^2}} - B e^{-\frac{|\mathbf{x}-\mathbf{x'}|^2}{b^2}} \tag{8}$$

4 Heterogeneous implementation

The first step is to partition the design into an hardware part and a software part. Since the goal is the meet the real-time constraints, this partition is based on the computational cost of each algorithmic component. For this, we evaluated the required throughput (in MB/s) for each algorithmic component. The results are reported in Tab. 1, for a full-HD resolution (720p) at 30 fps. Not surprisingly, the highest numbers are associated to the first two steps, which respectively computes the gradient and accumulates the histograms of their directions. The related computations involve massive, regular data-parallelism. They are therefore well suited to FPGA implementation. The last step (DNF filtering) exhibits a very moderate requirement in terms of data throughput but involves computations for which the accuracy is a key requirement. This step is therefore suited for a purely software CPU implementation. The two intermediate steps (normalization and SVM computation) have intermediate requirements both in terms of data throughput and accuracy. Because they require an tight coupling with the previous steps, with still significant throughput, they will benefit from an implementation on the FPGA.

The resulting proposal for the hardware-software heterogeneous architecture is sketched in Fig. 3. Mapping the three first steps on the same FPGA minimizes the risk of communication bottleneck between the hardware and software parts. On the other hand, having the post-filtering stage realized in software will make it easier to test and benchmark different post-filtering algorithms.

Function	Required throughput	Optimal target
Gradient	27.6 MB/s	FPGA
Histogram	27.6 MB/s	FPGA
Normalization	3.45 MB/s	FPGA
SVM	3.45 MB/s	FPGA
DNF	0.34 MB/s	CPU

Table 1: Required data throughput (720p resolution, 30 fps, 8x8 cells, 8 bins) and proposed target architecture

In the rest of this section, each design partition is further detailed and the associated implementation issues discussed. The complete system is then profiled and assessed in Sec. 5.

4.1 Gradient computation

Since the highest throughput appears right after pixel acquisition, the performance of this first stage significantly affects the overall results. Here, computation of the gradient is performed using the *HOG-Dot* algorithm proposed in [27]. With this algorithm, each gradient is discretized by projecting it onto a set of predefined directions θ_k ($k = 0 \dots N_b - 1$). The k^{th} projection is given by :

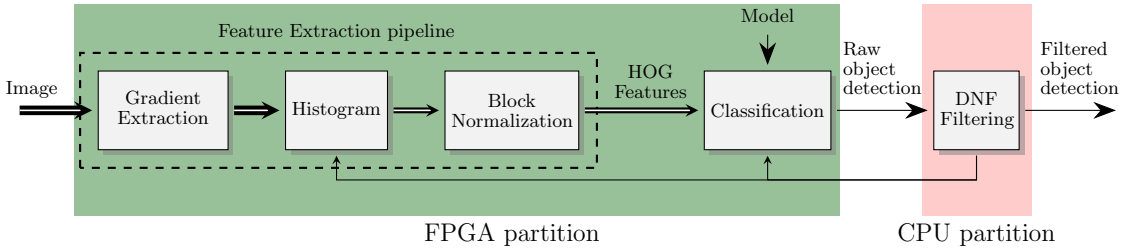


Fig. 3: Overview of the proposed hardware-software architecture.

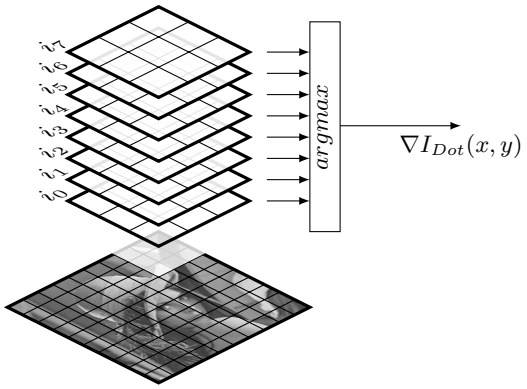


Fig. 4: The HOG-Dot parallel implementation.

$$\frac{\partial I}{\partial \hat{i_k}}(x,y) = \cos\theta_k \left[I(x+1,y) - I(x-1,y)\right] + \sin\theta_k \left[I(x,y+1) - I(x,y-1)\right] \quad (9)$$

where $\hat{i_k}$ is the unitary vector in direction θ_k.

The greatest projection then directly gives the closest gradient approximation in the discrete domain. The process is summarized in Fig. 4 (with $N_b = 8$), where ∇I_{Dot} here gives both to the discretized gradient orientation and the corresponding magnitude for each input pixel.

From an implementation point of view, this approach has two advantages. First, the N_b projections can be computed in parallel, thus significantly improving the overall throughput. Second, it involves only linear operations – and no square root neither arc tangent – and is therefore particularly suitable for hardware implementations. In fact, we have shown that, in this context, the use of linear approximations – compared to non-linear, floating point realizations – leads to an average error of less than 2% [26].

4.2 Histogram computation

Algorithmically speaking, this step is straightforward: it only consists in accumulating, in an array, each gradient orientation (the number N_b of discretized orientations therefore corresponds to the number of histogram bins). Nonetheless it raises challenging issues at the implementation level when applied to the data flow produced by the previous stage. Classically, histogram computation is decomposed in three successive steps: (i) address computation, (ii) memory read and addition (iii) memory write. With a synchronous implementation, the read-modify-write procedure therefore requires two clock cycles to handle each input data. The gradient computation stage, instead, produces one gradient direction per clock cycle. The architecture we used to solve this problem is depicted in Fig. 5. It is a modified version of one initially presented in [25] and uses two distinct histogram modules (SubCell0 and SubCell1 in Fig. 5). Input data is directed alternately to one or the other of these modules, so that none of them requires more than one access to the memory at each clock cycle. At the end of each cell, the final histogram is obtained by simply adding the two sub-histograms.

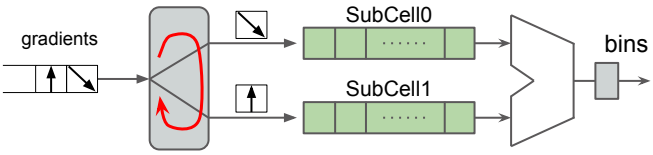


Fig. 5: Internal histogram cell generation.

4.3 Histogram Normalization

As previously underlined, a normalization step is required to limit the sensitivity of the algorithm to the variations in global luminance. We have evaluated well-known normalization schemes for HOG in terms of detection performance. Results are given in Fig. 6, using the Receiver Operating Characteristics (ROC) metric – *i.e.* by plotting the True Positive Rate (TPR) with respect to the False Positive Rate (FPR). With no surprise, a lack of normalization leads to the poorest performance, with a maximum of 83% of TPR at 10% FPR. The three other assessed normalization schemes (L1-norm, L1-srqt and

L2) give similar performances. We have chosen to implement the L1-norm because it offers the best trade-off between precision and implementation complexity. As shown in Fig. 6, the L1-norm shows only a limited loss in precision with respect to L2-norm – from 95% to 93% of TPR both evaluated at 10% FPR. Moreover, L1-norm does not require the square root operation with respect to L2-norm but only a division.

Our implementation of the L1-norm computation step is sketched in Fig. 7. The normalization factor is here computed as the sum of the histogram. A pre normalization factor is applied to each value to limit the effect of quantification due to fixed-point encoding. This factor can be adjusted according to the required accuracy on the one hand and the output data range on the other hand.

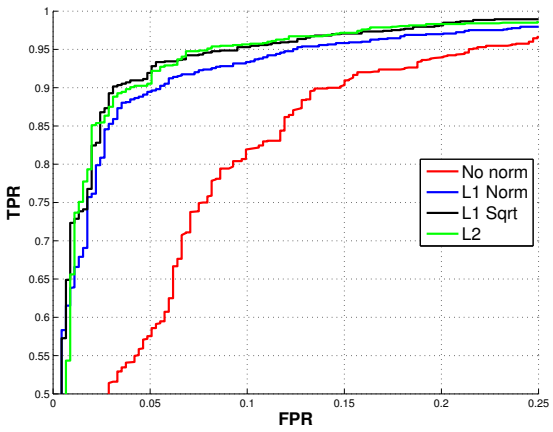


Fig. 6: Normalization output results.

In Fig. 7 a shift register (upper left) is used to buffer the histogram values belonging to the same cell so that each value can eventually be divided by the sum of values within his cell. This allows the normalization step to be carried out in a fully pipelined fashion. The division operation (right) is itself implemented using a pipelined architecture in order to maximize the global throughput.

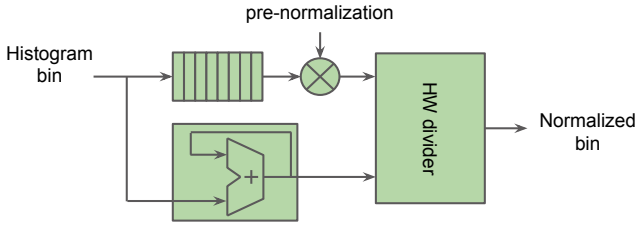
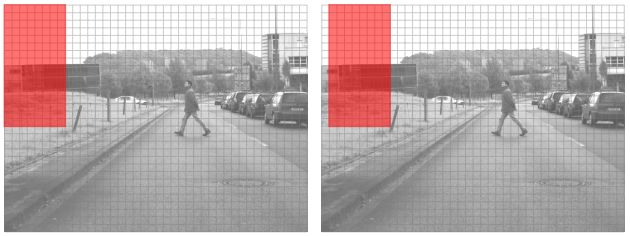


Fig. 7: L1 norm module.

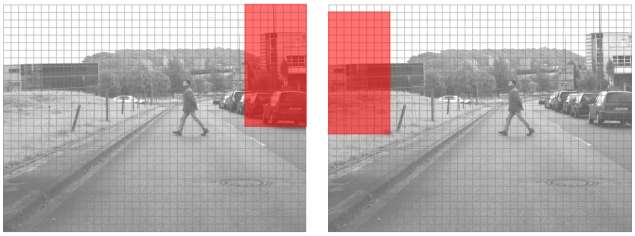
4.4 Support Vector Machine

As introduced in Eq. 5, the SVM classification is computed over image subsets, called *detection windows* (or simply *windows* in the sequel). Fig. 8, for example, shows a situation in which a detection window, here composed of 8×16 cells, is slided over the entire image in order to detect pedestrians.



(a) Initial Position

(b) Second window (one block shifted on the horizontal direction)



(c) Last window in the first row

(d) First window of the second row (one block shifted in the vertical direction)

Fig. 8: Illustration of the sliding detection window

In most implementations, this sliding mechanism is carried out sequentially, *i.e.* by processing one window after the other. This purely sequential approach is highly time consuming and, in practice practically precludes real-time performance to be obtained on general purpose CPUs. Our implementation circumvent this problem by leveraging the massive parallelism offered by FPGAs. The idea is to process the (overlapping) windows of a single row in parallel (e.g., Fig. 8 those located between positions a and c). For this, we introduce a dedicated accumulation unit called a *slice*. Each slice computes, in parallel, the dot product between the descriptors obtained on a set of neighboring windows and the model obtained by the offline training phase.

The dot product itself is obtained from the Eq. 3 with a linear kernel ($K(\mathbf{x_i}, \mathbf{x})$ is linear) and a binary

classification problem. For each window:

$$y_{n,m}(\mathbf{x}) = \sum_{\substack{1\leq i\leq 16\\ 1\leq j\leq 8}} \underline{w_B}_{i,j} \cdot \underline{x_B}_{n+i,m+j} \quad + b \tag{10}$$

with $n \in \{1, \cdots, N\}$ and $m \in \{1, \cdots, M\}$

where $\underline{x_B}_{n,m}$ is the HOG descriptor value at coordinates (n, m) and $y_{n,m}$ is the corresponding SVM detection result. As introduced in Eq. 5, N and M respectively gives number of windows in the vertical and horizontal direction.

Since weight coefficients have constant position within the detection window, each slice has a closed-loop mechanism for hardware reuse. Indeed, the first weight coefficients $\underline{w_B}_{1,j}$ for a block $\underline{x_B}_{n,m}$ are also the first coefficients for the next block $\underline{x_B}_{n+1,m}$ and so on. This temporal relationship allows the transfer of the weights between slices, thus reducing the memory footprint and wiring complexity of the design.

Slices are first structured into horizontal machines managing the horizontal overlap and the model weights distribution. For a specific line of HOG descriptors at the n vertical coordinate, the hardware instance H_z (as in Fig. 9) computes all the horizontal stride windows as follows:

$$\underline{y}_{n,1..M} = \left\{ \sum_{i=n}^{n+15} \sum_{k=j \bmod 8}^{j \bmod 8+7} \underline{w_B}_{\alpha,k} \cdot \underline{x_B}_{i,j} \middle| \begin{matrix} j \in \{1, ..., M\} \\ \alpha = i - n + 1 \end{matrix} \right\} \tag{11}$$

The horizontal machine computes the SVM projections by addressing the $w_{B\alpha,k}$ and $x_{Bi,j}$ values. The indexes (i, j) identifies the cell coordinates within the image. The horizontal iterations, relative to windows and to cells within the window are referenced with j and k respectively. The H_z machine finishes when the last cell in the $n + 15^t h$ cell row has been processed. In Fig. 9, the hardware architecture of the horizontal system is given for the generic line z. Here the SVM weights $\underline{w_B}_{\alpha,k}$ coming from the previous H_{z-1} flow as a carry chain among slices and then to the next H_{z+1} instance. In order to further minimize the number of concurrent read accesses to the memory containing the model descriptor, a circular chain of coefficients is deployed, allowing reuse of these coefficients between active H machines. Once the weight coefficients have been sent once, the machines internally explois the temporal correlations without requiring any further memory access.

The same approach is applied to a vertical overlap machine, as illustrated in Fig. 10, which shows how 16 horizontal H modules – as defined in the previous paragraph – are instantiated to process in parallel 16 lines of a set of windows, with the corresponding SVM weights distributed in a loop. The role of the output multiplexer

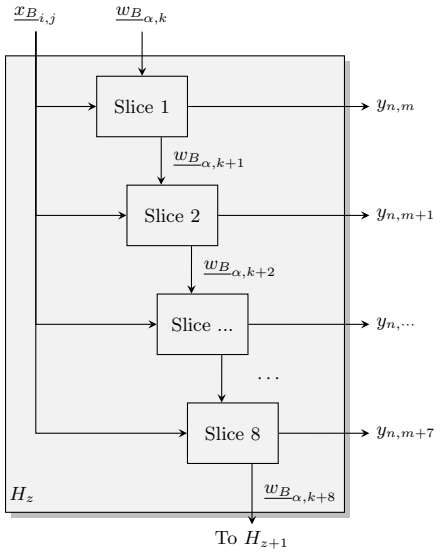


Fig. 9: The horizontal module H_z

is to select the SVM result corresponding to the corresponding window line, following equation Eq. 12. In this equation, $y_{n,m}$ is the result associated to the window at coordinates (n, m) and 16 is the number of lines per window.

$$\underline{y}_{n,1\cdots M} = \{H_z\}, \text{ with } z = n \bmod 16 \tag{12}$$

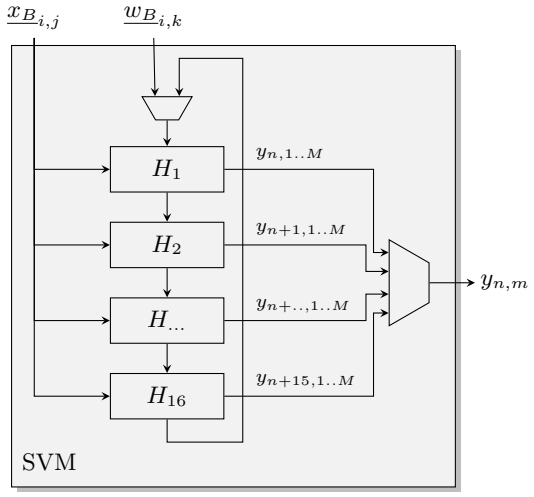


Fig. 10: The complete SVM architecture

The final resul is a parallel architecture capable of carrying out the SVM detections in a parallel manner.

4.5 Dynamic Neural Field

To make the DNF model simulation possible, discretization is performed in the spatial (grid lattice approximation) and temporal domains (Euler integration scheme). Using a matrix formulation, Eq. 6 becomes:

$$U(t+dt) = \left(1 - \frac{dt}{\tau}\right) U(t) + \frac{dt}{\tau}\left(C + \overline{Y}(t)\right) \quad (13)$$

where $\overline{Y}$ is obtained by first thresholding Y (from Eq. 5), and then normalizing it in $[0, 1]$. This operation has the effect of setting all values below $-\mu$ to 0, where μ corresponds to a tolerance on the negative side of the classification separator. Increasing μ (especially above the SVM margin value) leads to retaining a larger amount of false positive results. Yet since DNF models are designed to deal with low signal-to-noise ratios and effectively filter false positives out (i.e. limited FPR), setting $\mu > 0$ actually leads to higher TPR. Nevertheless, and in order to make a fair comparison with the raw data in the result section, μ was set to 0. The spatial competition term C is then defined by:

$$C = W^+ * \sigma(U(t)) - B \times \Sigma(U(t)) \quad (14)$$

where W^+ is the matrix version of the excitatory part of the kernel function w (see Eq. 8), and $\Sigma(U(t))$ is the grand sum of the matrix $\sigma(U(t))$. This approximation is made possible by setting $b = +\infty$, focusing on a single pedestrian at a time, but also limiting the convolution to a reduced excitatory kernel of 5×9. Other parameters take the following values: $A = 1$, $B = 0.3$, $a = 0.25$, assuming an affine transform is applied to the input space (matrix $\overline{Y}$ of 72×44) so that the pedestrian detection window (8×16 HoG cells) corresponds to a unit square.

5 Experimental results

In order to evaluate the heterogeneous architecture proposed in the previous section, a prototype implementation has been developed. This prototype is built upon an Arrow SOC development board. This board embeds an Altera Cyclone V FPGA with up to 45k Arithmetic Logic Unit (ALM) and 336 9x9 DSP multipliers. Within the FPGA fabric, a dual core ARM Cortex-A9 is deployed as a Hard Processor System (HPS). The HPS and the FPGA part are tightly coupled through the AMBA bus, which allows interoperable communications between hardware and software routines.

Two issues are addressed by the evaluation process. Performance on the one hand and reliability on the other hand. Performance is assessed in terms of processing time and hardware resource usage (Sec. 5.1). Reliability is assessed in terms of detection accuracy with respect to state of art existing realizations with different evaluation conditions (Sec. 5.2).

5.1 Performance evaluation

Tab. 2 gives the details of the FPGA implementation as obtained using the Altera Quartus II 13.1 toolchain. The reported values give, for each algorithmic step the resource usage in terms of ALM, RAM and DSP, the latency (L) and the estimated maximum clock frequency.

The complete processing pipeline takes only 18% of the available hardware resources. The required amount of memory (for line buffering and internal data storage), in particular, is less than 200 Kbit and largely fits in the available on-chip FPGA RAM.

Concerning latency, it is defined as the time between the consumption of the first input data and the production of the first output result. This time directly depends on the depth of the processing pipeline. In our case, in does not depend on the value of the input data and is therefore constant.

	ALMs	RAM	DSPs	L	f_{MAX}
Gradient	464	16 Kb	8	10	107
Histogram	244	131 Kb	0	5120	145
Norm	400	8 K	0	16	83.9
SVM	7300	42 Kb	128	10240	130

Table 2: FPGA hardware results for VGA resolution. The f_{MAX} and Latency (L) are expressed as MHz and clock cycles respectively.

In order to demonstrate the potential benefits of our heterogeneous architecture, the results given above have been compared to those obtained with two other architectures: a commercial Intel i7-870 workstation and an embedded ARM HPS subsystem. For the latters, a state of the art OpenCV implementation of the HOG and SVM algorithms was used. Results are given in Tab. 3. Because the notion of clock frequency is not directly applicable to software implementations, comparison is performed in terms of processing time for a single frame. It must be noticed, however, that this notion of processing time has an interpretation which ultimately depends on the platform. For a purely software implementation, it is generally defined as the interval separating the end of the input frame acquisition and the end of the processing of this frame by the CPU ((b)-(d) in Fig. 11). For a FPGA-based implementation, processing actually occurs in parallel with acquisition, in a fully pipelined fashion. Processing time, in this case, can be evaluated by measuring the latency of the output result (interval (b)-(c) or (a)-0 in Fig. 11).

In our case, the maximum clock frequency and latencies reported in Tab. 2 lead to a global latency of 184 μs (sum of each module latency). Compared to the I7-870 and ARM implementations this corresponds to a reduction of 400x and 4630x respectively. In terms of FPS, the

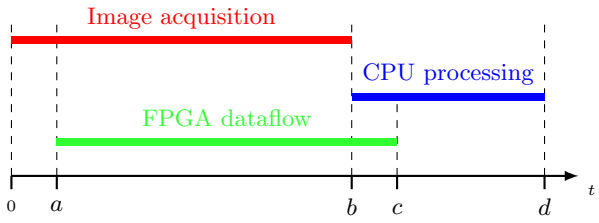


Fig. 11: Performance comparison.

multiplication factor, with respect to these same implementations, are of 20x and 218x respectively. However, these numbers should be interpreted keeping in mind that the reported processing times for the CPU implementations ((b)-(d) in Fig. 11) do not take into account the image acquisition time. This might lead to a significant overestimation of the performance results. For the FPGA implementation instead, the fps value is experimentally measured and only depends to the input pixel rate. The maximum clock frequency, which in our case, gives the maximum pixel rate, is 83.9 MHz. For VGA resolution, this gives a maximum throughput of 273 fps.

	i7 870	ARM	FPGA
Function	time (μs)	time (μs)	time (μs)
Gradient	38200	425756	0.12
Histogram			61.4
Normalization			0.2
SVM	36000	376566	122
Overall HOG	74200	802322	184
fps	13.5	1.25	273

Table 3: Comparison of processing times at VGA resolution.

We also compared our implementation to other FPGA-based implementations of the HOG-SVM application. In [19] a solution with 10 parallel processing elements is described. The reported maximum clock frequency is 127.49 MHz but with a maximum frame rate of 30 fps at VGA resolution. Another heterogeneous HOG architecture has been proposed in [4]. But, due to the processing bottleneck caused by the PCIe communication interface between the FPGA and the rest of the system, the histogram computation is 10x time slower than in our solution (657 μs instead of 64.4 μs). In [11] the PCIe limitation is significantly reduced to 150 μs with Full-HD images sent through a GigaE connection but still represents a significant bottleneck, hindering global system performance for the CPU-FPGA proposed solution. In [1] an hybrid FPGA-CPU-GPU solution is described. Using FPGA modules leads to a processing time of 311 μs only for gradient and histogram computations. The normalization step and the Gaussian kernel SVM are then handled by an external CPU-GPU device though multiple DMA instances.

Compared to those realizations, our architecture minimizes bottlenecks by privileging on-chip communications and limiting inter-chip communications to low bandwidth data. Implementing all the most throughput demanding modules in the FPGA, in particular, allows these throughputs to be limited only by the critical path in the associated circuitry and not by the bandwith of the hardware-software interface. The proposed hybrid approach minimizes also the external interfaces requirements thus reducing the global system complexity.

5.2 Detection accuracy

In this section we evaluate the final accuracy of our detection application by comparing it to other state of the art similar applications. The selected alternatives are those described in [11], [19], [1] and discussed in Sec. 2.

Evaluation is carried out in two steps : first with the well-known INRIA dataset, second with real-world VGA image sequences taken from the Daimler Pedestrian Benchmark Dataset [7]. In both cases, the classifiers were trained with positive and negative image samples from the INRIA pedestrian dataset [5], with an offline training phase performed with the SVMLight framework [18].

5.2.1 Evaluation using the INRIA dataset

Results for this first evaluation step are given in Tab. 4. For fair comparisons, all the considered techniques are trained with the INRIA train dataset [5] and verified with the INRIA test subset (1126 positives and 453 negatives 64×128 windows) and no post-SVM filtering was used. For [1] only the gradient and histogram parts have been ported inside the FPGA.

All implementations exhibit comparable results in terms of TPR (the most critical aspect for pedestrian detection applications). Our implementation exhibit a slightly higher FPR (4%). This effect has been tracked down to the use of the L1-norm approximation and can be eliminated by adding a post-SVM filtering stage (as shown in the next section).

	[11]	[19]	[1]	Our
TPR	93 %	95 %	95.4 %	94.9 %
FPR	1.0 %	1.0 %	0.1 %	5.5 %

Table 4: Comparisons with state of the art FPGA HOG implementations on the INRIA train/test dataset.

.

	[3] CPU	[7] CPU T_p=2.5s		[7] CPU T_p=250ms		Our FPGA+ARM	
	Raw	Raw	Tracking	Raw	Tracking	Raw	DNF
TPR	41.5	64.3	68.7	67.4	79.1	91.0	80.3
FPR	-	1.17	0.14	14.3	1.3	17.6	1.0
AUC	-	-	-	-	-	0.785	0.891

Table 5: Detection performance for cross-dataset implementations. Area Under ROC Curve (AUC) $\in [0, 1]$, True Positive Rate (TPR) and False Positive Rate (FPR) in percents.

5.2.2 Evaluation using the real world video sequences

Obviously, and as recalled in [3], this step is mandatory for any pedestrian detection system aiming at realistic applications, such as those integrated in ADAS. We therefore have evaluated our application using a dataset consisting in sequences of VGA images obtained with a camera mounted on an outside moving vehicle (Daimler pedestrian video sequence [7]). This kind of data is indeed the ideal benchmark for an ADAS prototype because experimental evaluation here nearly perfectly matches the actual exploitation conditions (with different intrinsic camera parameters, extreme luminance variation and rapid scene changes). Moreover since most of the false positive detections are not temporally coherent, they can be removed using a DNF-based post-processing step (as showed below). Training of the models, however, is still be carried out using the INRIA dataset. This so-called *cross-dataset* approach has been proposed in [7] and [3]. As a proof-of-concept, a single window scale is considered in this work. Although it substantially reduces the detection performance with respect to a multi-scale implementation, it represents a benchmark to validate our methodology.

Results are given in Tab. 5 using three metrics : the TPR/FPR metrics on the one hand and the AUC metric on the other hand. The former is the same as previously defined. The latter has been proposed in [6] in order to evaluate the overlap between the detected and the "ground truth" bounding boxes (BB_{dt} and BB_{gt} in the sequel) when multiple detection windows are placed around a pedestrian silhouette. For this, a overlaping factor is computed as

$$a = \frac{Area(BB_{dt} \cap BB_{gt})}{Area(BB_{dt} \cup BB_{gt})} \tag{15}$$

and compared to given threshold a_0. In the PASCAL challenge [8], a_0 is set to 0.5 and we adopt it for comparisons as well. Each detected bounding box is evaluated with respect to the manually labelled ground truth. The higher the overlap, the higher the detection confidence results. If the overlap does not exceed the threshold value, the bounding box is labeled as false detection. The final detection performance is finally expressed as the Area Under Curve (AUC) and the TPR/FPR with respect to the "ground truth" bounding box BB_{gt}. Receiver Operating characteristic (ROC) curves are generated by varying the threshold value and then used to compute the AUC metric. The AUC metrics is 1 is for the ground truth, 0 for a fully wrong result and 0.5 for chance level. Because it does not depend on the detection threshold (b parameter in Eq. 10), this metric is particularly useful to estimate the detection gain with respect to brute FPR/TPR values.

In Tab. 5, the *Raw* columns correspond to the zero-crossing output of the SVM with respect to b (e.g., positive values as pedestrians), while Tracking and DNF results are relative to the post-processing method applied. Note that [3] reports only TPR number, without any additional implementation detail. Results in [7] are given for two implementations based on different constraints on a 2.66 GHz Intel processor. Note also that these two implementations differ in the size of the detection grid applied to the image and thus exhibit distinct processing times (with respect to trajectory-based detections [10]).

First of all, Tab. 5 clearly shows that overall performances, compared to those given in Tab. 4, are significantly reduced. This is an unavoidable consequence of using a *cross-dataset*-based approach, in which the detector is confronted to unexpected features. Moreover, raw SVM detections are considerably less accurate than filtered ones. The work reported in [7] shows that temporal tracking allows a reduction of the FPR from 1.17% to 0.14% and from 14.3%0 to 1.3% with 2.5 s and 250 ms implementations respectively and an increase of the TPR from 64.3% to 68.7% and from 67.4% to 79.1% respectively. With respect to the TPR, our solution outperforms other implementations regardless of the post processing stage, and leads to a reduced miss rate for pedestrians in a real deployment scenario.

As already shown in Tab. 4, our architecture presents an higher raw FPR with respect to other methods. Indeed, with cross-dataset validation the raw FPR reaches 17.6% with respect to the 5.5% achieved with INRIA dataset. This rate is also higher than the other considered cross-dataset validations, which limits the FPR to 1.17% and 14.3% respectively. Nevertheless, our solution performs significantly better for raw TPR values, reaching 91% of accuracy with respect to 41.5%, 64.3% and 67.4% of the considered comparison in Tab. 5.

Tab. 5 also shows the positive effect of the post-processing stages, either based on tracking or on DNF.

For our implementation, in particular, the DNF step drastically reduces the number of false detections, reducing the FPR from 17.6% to 1.0% (82% improvement). This drastic reduction in FPR comes at a price though. As shown in Tab. 4, the TPR value results are indeed slightly reduced by the DNF post-processing (-10%). This effect is due to some misled false detection suppression, which are particularly important in case of not completely overlapping detection windows. Despite the TPR degradation, the AUC metric shows that DNF has improved the system response. Indeed, the FPR improvements largely compensates for the TPR degradation, and the resulting SVM+DNF combination globally improves the detection reliability. Fig. 12 illustrates visually the impact of the DNF-based post-processing stage on the detection results. The left column (Fig.12a, 12c, 12e, 12g) shows the raw SVM detections provided by the hardware system on four consecutive frames (from Daimler pedestrian dataset). The right column (Fig. 12b, 12d, 12f, 12h) gives to the detections after the DNF filtering stage has been applied. As suggested by results in Tab. 5, the FPR value has been considerably reduced with a stable and consistent target detection.

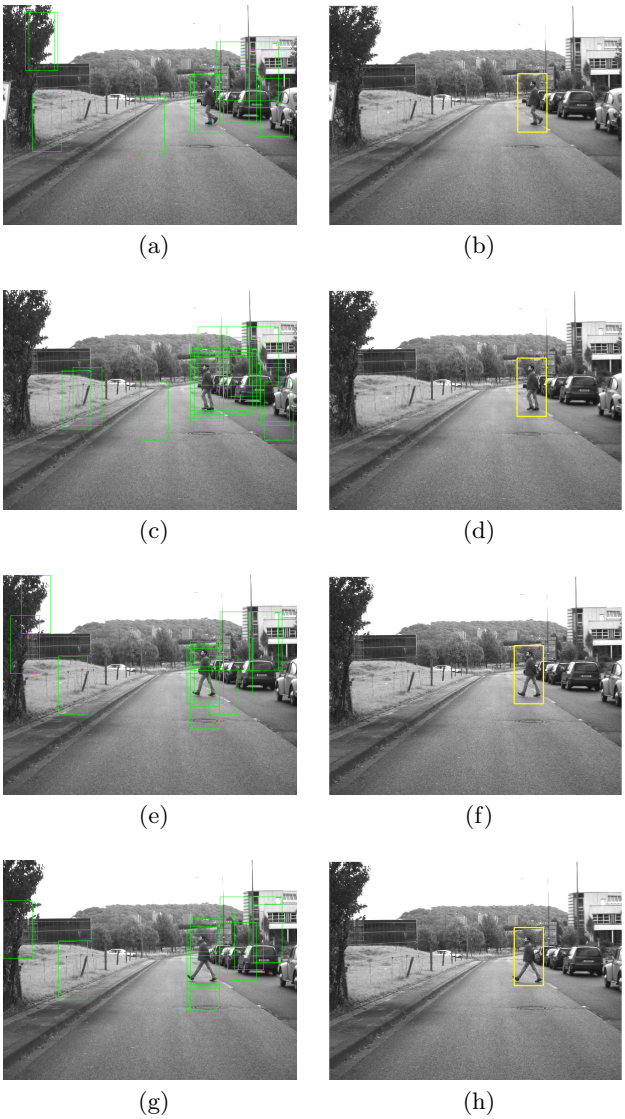


Fig. 12: Raw SVM detections (left) and DNF filtering results (right) computed by the proposed architecture. Subsequent frames were chosen to display the spatiotemporal coherence of the SVM results.

6 Conclusions

In this paper, a bio-inspired heterogeneous architecture has been presented aiming at meeting the requirements of real-time pedestrian detection applications.

Experimental results, obtained in realistic conditions, show that this architecture achieves satisfactory detection performance for this kind of applications, thus envisioning a possible ADAS deployment. They show in particular the benefit of a co-design partitioning methodology, in which a tight coupling of hardware and software processing modules leads both to a gain in performance and an improvement in detection reliability. They also demonstrate the benefits of a spatio-temporal DNF-based post-filtering step in terms of detection reliability, by reducing the amount of false positive detections.

These results also suggest several possible improvements. First, with respect to the hardware implementation, an improved normalization schema should be included to improve the robustness against variation in global luminance. Second, using a multi-scale approach in each detection window is likely to improve the detector sensitivity and the DNF post-processing as well. Indeed, our prototype considers only one detection window scale at a time. That is obviously a limitation and it is reducing the TPR with respect to what a multi-scale implementation would do. Moreover, a multi-scale implementation would especially benefit from a heterogeneous architecture by leveraging on parallel hardware processing pipelines and a final high level data fusion step. Third, the DNF post-processing stage, now implemented in software, could be moved to hardware. Efficient FPGA implementations of DNFs have been recently proposed [32] demonstrating that, though complex, hardware implementations of this kind of algorithm are indeed possible and profitable. Such an optimisation would allow the introduction of more complex, sparse model based, filtering mechanisms on the CPU partition, thus pushing again further the complementarity of parallel and sequential capabilities of an heterogeneous architecture.

Finally, another aspect needing further investigation is power consumption, an issue whose importance is likely to become crucial for real, embedded ADAS deployment.

Acknowledgment

This work has been sponsored by the French government research programme "Investissements d'avenir" through the IMobS3 Laboratory of Excellence (ANR-10-LABX-16-01), by the European Union through the program Regional competitiveness and employment 2007-2013 (ERDF Auvergne region), and by the Auvergne region.

References

1. Bauer S, Köhler S, Doll K, Brunsmann U (2010) FPGA-GPU architecture for kernel SVM pedestrian detection. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, IEEE, pp 61–68
2. Benenson R, Mathias M, Timofte R, Van Gool L (2012) Pedestrian detection at 100 frames per second. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, pp 2903–2910
3. Benenson R, Omran M, Hosang J, Schiele B (2014) Ten years of pedestrian detection, what have we learned? In: Computer Vision-ECCV 2014 Workshops, Springer, pp 613–627
4. Blair C, Robertson NM, Hume D (2013) Characterizing a heterogeneous system for person detection in video using histograms of oriented gradients: Power versus speed versus accuracy. Emerging and Selected Topics in Circuits and Systems, IEEE Journal on 3(2):236–247
5. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol 1, pp 886–893
6. Dollar P, Wojek C, Schiele B, Perona P (2012) Pedestrian detection: An evaluation of the state of the art. Pattern Analysis and Machine Intelligence, IEEE Transactions on 34(4):743–761
7. Enzweiler M, Gavrila DM (2009) Monocular pedestrian detection: Survey and experiments. Pattern Analysis and Machine Intelligence, IEEE Transactions on 31(12):2179–2195
8. Everingham M, Van Gool L, Williams CK, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. International journal of computer vision 88(2):303–338
9. Gandhi T, Trivedi M (2007) Pedestrian protection systems: Issues, survey, and challenges. Intelligent Transportation Systems, IEEE Transactions on pp 413–430
10. Gavrila DM, Munder S (2007) Multi-cue pedestrian detection and tracking from a moving vehicle. International journal of computer vision 73(1):41–59
11. Hahnle M, Saxen F, Hisung M, Brunsmann U, Doll K (2013) FPGA-based real-time pedestrian detection on high-resolution images. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on, pp 629–635
12. Happe M, Lübbers E, Platzner M (2013) A self-adaptive heterogeneous multi-core architecture for embedded real-time video object tracking. Journal of real-time image processing 8(1):95–110
13. Hartmann C, Yupatova A, Reichenbach M, Fey D, German R (2015) A holistic approach for modeling and synthesis of image processing applications for heterogeneous computing architectures
14. Hong GS, Kim BG, Hwang YS, Kwon KK (2015) Fast multi-feature pedestrian detection algorithm based on histogram of oriented gradient using discrete wavelet transform. Multimedia Tools and Applications
15. Hua C, Makihara Y, Yagi Y, Iwasaki S, Miyagawa K, Li B (2015) Onboard monocular pedestrian detection by combining spatio-temporal HOG with structure from motion algorithm. Machine Vision and Applications 26(2-3):161–183
16. Hussein M, Porikli F, Davis L (2009) A comprehensive evaluation framework and a comparative study for human detectors. Intelligent Transportation Systems, IEEE Transactions on 10(3):417–427
17. Inggs G, Thomas DB, Luk W (2015) An efficient, automatic approach to high performance heterogeneous computing. arXiv preprint arXiv:150504417
18. Joachims T (1999) Svmlight: Support vector machine. http://svmlight.joachims.org/
19. Kadota R, Sugano H, Hiromoto M, Ochi H, Miyamoto R, Nakamura Y (2009) Hardware architecture for hog feature extraction. In: Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on, pp 1330–1333
20. Kim SH, Kim JS, Wan V, Suh IH (2015) Automotive adas camera system configuration using multi-core microcontroller. Tech. rep., SAE Technical Paper
21. Klaser A, Marszałek M, Schmid C (2008) A spatio-temporal descriptor based on 3d-gradients. In: BMVC 2008-19th British Machine Vision Conference, British Machine Vision Association, pp 275–1
22. Liu W, Yu B, Duan C, Chai L, Yuan H, Zhao H (2015) A pedestrian-detection method based on heterogeneous features and ensemble of multi-view–pose parts. Intelligent Transportation Systems, IEEE Transactions on 16(2):813–824
23. Ma X, Najjar W, Roy-Chowdhury A (2015) Evaluation and acceleration of high-throughput fixed-point object detection on FPGAs. Circuits and Systems for Video Technology, IEEE Transactions on 25(6):1051–1062
24. Machida T, Naito T (2011) GPU & CPU cooperative accelerated pedestrian and vehicle detection. In: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, pp

506–513
25. Maggiani L, Salvadori C, Petracca M, Pagano P, Saletti R (2014) Reconfigurable architecture for computing histograms in real-time tailored to FPGA-based smart camera. In: Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on, IEEE, pp 1042–1046
26. Maggiani L, Bourrasset C, Berry F, Sérot J, Petracca M, Salvadori C (2015) Parallel image gradient extraction core for FPGA-based smart cameras. In: Proceedings of the 9th International Conference on Distributed Smart Camera, ACM, pp 128–133
27. Maggiani L, Bourrasset C, Petracca M, Berry F, Pagano P, Salvadori C (2015) HOG-Dot: A parallel kernel-based gradient extraction for embedded image processing. Signal Processing Letters, IEEE 22(11):2132–2136
28. Ojala T, Pietikäinen M, Harwood D (1996) A comparative study of texture measures with classification based on featured distributions. Pattern recognition 29(1):51–59
29. Quinton JC, Girau B (2011) Predictive neural fields for improved tracking and attentional properties. In: Neural Networks (IJCNN), The 2011 International Joint Conference on, IEEE, pp 1629–1636
30. Reiche O, Haublein K, Reichenbach M, Hannig F, Teich J, Fey D (2015) Automatic optimization of hardware accelerators for image processing. In: DATE Workshop on Heterogeneous Architectures and Design Methods for Embedded Image Systems, HIS 2015
31. Tanabe J, Toru S, Yamada Y, Watanabe T, Okumura M, Nishiyama M, Nomura T, Oma K, Sato N, Banno M, et al (2015) 18.2 a 1.9 tops and 564gops/w heterogeneous multicore soc with color-based object classification accelerator for image-recognition applications. In: Solid-State Circuits Conference-(ISSCC), 2015 IEEE International, IEEE, pp 1–3
32. Vangel BCD, Torres-Huitzil C, Girau B (2015) Randomly spiking dynamic neural fields. ACM Journal on Emerging Technologies in Computing Systems (JETC) 11(4):37
33. Vapnik V (1982) Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA
34. World Health Organization (2013) Road traffic injuries. http://www.who.int/mediacentre/factsheets/fs358/en/
35. Wu S, Laganière R, Payeur P (2015) Improving pedestrian detection with selective gradient self-similarity feature. Pattern Recognition 48(8):2364–2376
36. Yadav R, Senthamilarasu V, Kutty K, Vaidya V, Ugale S (2015) A review on day-time pedestrian detection. Tech. rep., SAE Technical Paper
37. Yao S, Pan S, Wang T, Zheng C, Shen W, Chong Y (2015) A new pedestrian detection method based on combined HOG and LSS features. Neurocomputing 151:1006–1014
38. Zhang J, Huang K, Yu Y, Tan T (2011) Boosted local structured hog-lbp for object localization. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, pp 1393–1400