J Sign Process Syst (2016) 84:325–337
DOI 10.1007/s11265-015-1080-7

An Energy-Efficient Hardware Implementation of HOG-Based Object Detection at 1080HD 60 fps with Multi-Scale Support

Amr Suleiman[1] · **Vivienne Sze**[1]

Received: 12 January 2015 / Revised: 21 July 2015 / Accepted: 30 October 2015 / Published online: 4 December 2015

Abstract A real-time and energy-efficient multi-scale object detector hardware implementation is presented in this paper. Detection is done using Histogram of Oriented Gradients (HOG) features and Support Vector Machine (SVM) classification. Multi-scale detection is essential for robust and practical applications to detect objects of different sizes. Parallel detectors with balanced workload are used to increase the throughput, enabling voltage scaling and energy consumption reduction. Image pre-processing is also introduced to further reduce power and area costs of the image scales generation. This design can operate on high definition 1080HD video at 60 fps in real-time with a clock rate of 270 MHz, and consumes 45.3 mW (0.36 nJ/pixel) based on post-layout simulations. The ASIC has an area of 490 kgates and 0.538 Mbit on-chip memory in a 45 nm SOI CMOS process.

Keywords Object detection · Histogram of oriented gradients · Multi-scale · Low power architectures · Embedded vision

1 Introduction

Object detection is needed for many embedded vision applications including surveillance, advanced driver assistance systems (ADAS) [1], portable electronics and robotics. For these applications and others, it is desirable for object detection to be real-time at high frame rates, robust and energy-efficient. Real-time processing is necessary for applications such as ADAS, and autonomous control in unmanned aerial vehicles (UAV), where the vehicle needs to react quickly to fast changing environments. High frame rate enables *faster* detection to allow more time for course correction. For detection robustness, it is essential that detectors support multiple image scales to detect objects with variable sizes. As shown in Fig. 1, the size difference can be due to different distances from the camera (i.e. an object's height is inversely proportional to its distance from the camera [2]), or due to the actual size of the object (e.g. pedestrians with different heights). In addition, high resolution images, such as high definition (HD), enable *early* detection by having enough pixels to identify far objects, which is particularly important for fast moving objects. Finally, for energy consumption, in both UAV and portable electronics, the available energy is limited by the battery whose weight and size must be kept to a minimum [3]. Additionally, heat dissipation is a crucial factor for ADAS application [4].

A conventional method of object detection involves translating the image from pixel space into a higher dimensional feature space. Classification is then used at different regions in the image to decide whether a specific object exists or not. Histogram of Oriented Gradients (HOG), which looks at the distribution of edges, is a widely accepted feature for object detection [6]. HOG features provide a reasonable trade-off between detection accuracy and complexity compared to alternative richer features [7]. A combination of other types of features can be added to HOG for more accurate detection with the cost of more computation power. However, no single feature has been shown to outperform

✉ Amr Suleiman
suleiman@mit.edu

Vivienne Sze
sze@mit.edu

[1] Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 77 Massachusetts Ave, room 38-260, Cambridge, MA 02139, USA

Figure 1 Examples of pedestrians with different sizes based on their distances from the camera and/or their different heights. Images are taken from INRIA person dataset [5].

HOG, which makes HOG-based detection the baseline for object detection systems [2].

In this paper, we describe a hardware-friendly real-time energy-efficient HOG algorithm for object detection, with multi-scale support. The resulting implementation delivers high-throughput processing to achieve real-time, robust and accurate object detection at high frame rates with low hardware and energy costs. The main contributions of this work are:

1. Efficient scale selection and generation for multi-scale detection.
2. Parallel detectors with voltage scaling.
3. Image pre-processing to reduce multi-scale memory and processing overhead.

The rest of the paper is organized as follows. Section 2 provides a survey of existing HOG-based object detector implementations. Section 3 describes the HOG detection algorithm and the importance of multi-scale detection for robustness. Section 4 describes the hardware architecture of the HOG-based object detector. Section 5 introduces the parallel detectors architecture and voltage scaling. Then, image pre-processing is introduced in Section 6. We present the performance metrics including detection accuracy and hardware complexity in Section 7 and conclude in Section 8.

2 Previous Work

The majority of the published implementations of HOG-based object detection are on CPU and GPU platforms. In addition to consuming power in hundreds of Watts (e.g., Nvidia 8800 GTX GPU consumes 185W [8]), which is not suitable for embedded applications, they often cannot reach high definition (HD) resolutions. Authors in [9] demonstrate a real-time object detection system with multi-scale support on a CPU processing 320×240 pixels at 25 frames per second (fps). The implementation in [10] achieves higher throughput on a GPU at 100 fps using the approach presented in [7] to speed up feature extraction, but with a resolution of 640×480 pixels.

For higher throughput, FPGA-based implementations have recently been reported. Different parts of the detector are implemented in [11] on different platforms: HOG feature extraction is divided between an FPGA and a CPU, and SVM classification is done on a GPU. It can process 800×600 pixels at 10 fps for single scale detection. The

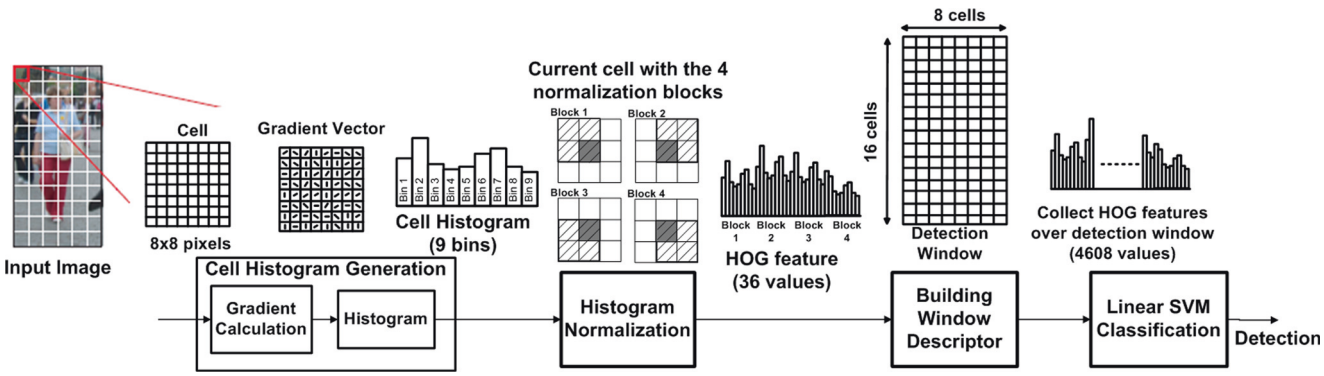

Figure 2 Object detection algorithm using HOG features.

entire HOG-based detector is implemented in [12] on an FPGA, and can process 1080HD video (1920×1080 pixels) at 30fps. However, the implementation only supports a single image scale. An ASIC version of this design is presented in [13], with dual cores to enable voltage scaling for power consumption of 40.3mW for 1080HD video at 30fps, but still only supports a single image scale. A multi-scale object detector is demonstrated on an FPGA in [14] that can process 1080HD at 64 fps. The 18 scales are time-multiplexed across 3 successive frames. As a result, effectively only 6 scales are processed per frame. It should be noted that these hardware implementations have relatively large on-chip memory sizes (e.g., [13] uses 1.22 Mbit on ASIC, [14] uses 7 Mbit on FPGA), which contributes to increased hardware cost.

Thus, from the previous discussion, none of the existing implementations satisfies all the desired requirements for accurate and robust object detection in embedded systems, which include real-time, high resolution (1080HD), high frame rate (>30 fps), multiple image scale, low power and low hardware costs.

3 Overview of the HOG Algorithm

Figure 2 shows a block diagram of the steps involved in object detection using HOG features as presented in [6]. The image is divided into non-overlapping 8×8 pixels patches called *cells*, where gradients are calculated at each pixel. A histogram of the gradient orientations with 9 bins is generated for each cell. The histogram is then normalized by its neighboring cells to increase robustness to texture and illumination variation. For each cell, the normalization is performed across *blocks* of 2×2 cells resulting in a final 36-D HOG feature vector.

The HOG feature vector is extracted for cells in a predefined detection window. In this work, the detection window is chosen to be 128×64 pixels which is suitable for pedestrian detection [6]. This window sweeps the entire image with non-overlapping cells. A conventional way to perform detection is by training a support vector machine

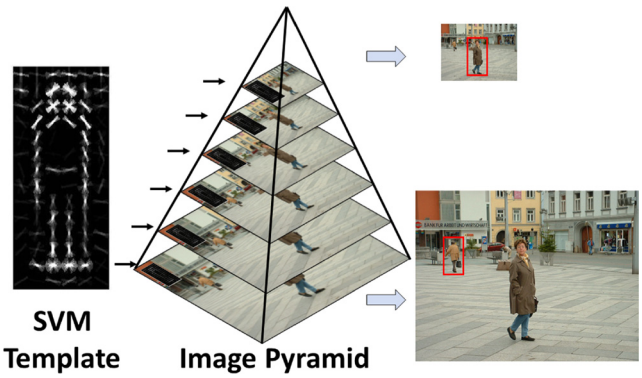

Figure 3 Image pyramid with multiple scales. Processing all scales with the same SVM template can detect objects of different sizes.

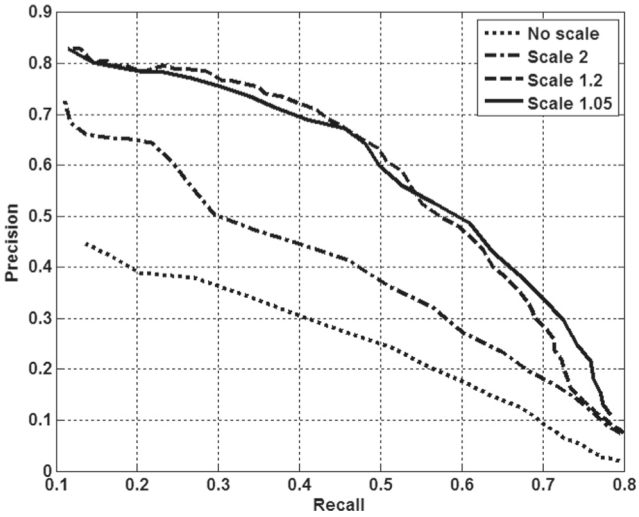

Figure 4 Precision-Recall curves for INRIA person dataset [5] using different scaling factors: single scale (AP=0.166), scale factor of 2 (AP=0.275), scale factor of 1.2 (AP=0.391), and scale factor of 1.05 (AP=0.401, used in the original HOG algorithm [6]).

Figure 5 Scanning order in reading the pixels from a frame (1080HD for this example). **a** Row raster scan. **b** Column raster scan.

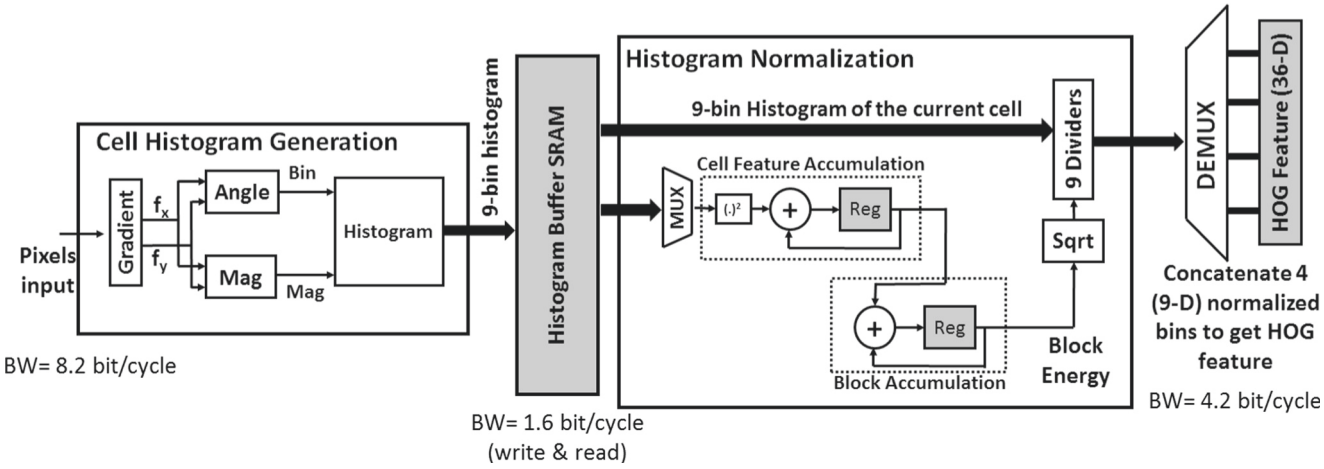

Figure 6 Block diagram of HOG feature extraction with average input and output bandwidths.

(SVM) [15] classifier for the predefined detection window size. The classifier has the same dimension as the detection window. The classification output is referred to as a score, and is compared to a threshold to make a detection decision. This operation is repeated as the detection window sweeps the image.

As mentioned in Section 1, scaling is an important factor in object detection algorithms since there is no prior knowledge about object size/distance from camera. Generally, and since most of the features are not scale invariant, there are two methods to perform multi-scale detection. The first *naive* approach is to have multiple SVM classifiers, each trained for the same object at a different size. In this case, the weights of the different SVM classifiers must be determined and a unique detector is used for each size. This approach is rarely used because it increases the complexity of the training process where multiple classifiers have to be trained. In addition to that, and from the hardware point of view, these classifiers coefficients have to be stored on-chip and it is shown later in this work that the SVM weights memory consumes a significant amount of power.

The second approach, which is conventionally used in object detectors [2] and is carried out in this work, is to have only one SVM classifier for one detection window size, and to generate an image pyramid composed of multiple scaled versions of the same frame, which is then processed by the same detector as shown in Fig. 3. In this approach, small objects can be detected in the high resolution scales while large objects can be detected in the low resolution scales, all with the same classifier. The ratio between the dimensions of successive scales in the image pyramid is called the *scale factor*.

The precision-recall curve shown in Fig. 4 is one method to measure the detection accuracy. Increasing the number of processed scales, by reducing the scale factor, increases the Average Precision (AP)[1] from 0.166 with single scale to 0.401 with scale factor of 1.05 (44 scales per 1080HD frame). However, this comes at the cost of increased computation in terms of generating the image pyramid and processing the newly generated pixels for each scale.

4 Hardware Architecture

In this work, a cell-based approach is used where one cell is processed at each stage of the architecture. There are various orders in which the cells of a 2-D image can be processed by the object detection hardware. The following two orders were examined as shown in Fig. 5: raster scan in row order and raster scan in column order. While the processing order does not affect the detection accuracy, it does impact the size of the on-chip memory in the hardware. Note that the computation pipeline of the object detection core, consisting of the detector and the scale generator, is the same for both vertical and horizontal scans; only the memory controller and memory sizes change. In the following sections, an architecture based on column raster scan will be discussed. In Section 7.3, a detailed hardware comparison between the two approaches is presented.

4.1 Detector

The detector can be divided into HOG feature extraction and SVM classification. The feature extraction shown in Fig. 6 includes cell histogram generation, the histogram buffer, and histogram normalization. The SVM classification shown in Fig. 9 includes multiply-and-accumulate

[1] Average precision measures the area under precision recall curve. Higher average precision means better detection accuracy.

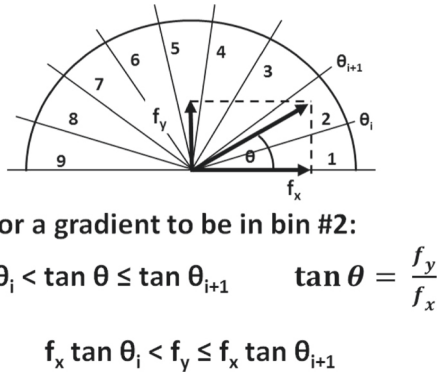

Figure 7 Histogram bins can be calculated without the actual value of the gradient angle. In the shown example, the pixel is assigned to bin 2 if the shown inequality has been satisfied. Note that (θ_i) is constant for all i's.

(MAC) units, the accumulation buffer, and the SVM weights buffer.

4.1.1 HOG Feature Extraction

Figure 6 shows the block diagram of the HOG feature extraction unit. It is composed of two key blocks: cell histogram generation and histogram normalization. In the cell histogram generation, a gradient filter [-1 0 1] is used to generate a pair of horizontal and vertical gradients at each pixel in the 8×8 input cell[2]. The orientation and the magnitude of the gradient are then calculated from this pair, and a histogram of 9 bins is generated for the cell. To reduce the implementation cost, the following approximations were performed:

- *Orientation*: We carried out the orientation binning scheme similar to [11, 16]. As the orientation is only used to choose the histogram bin, the actual angle of the gradient does not need to be calculated. The limit angles for each bin are known constants (θ_i), as shown in Fig. 7. Each gradient bin can be calculated using constant multiplications rather than complex computation to calculate the actual gradient angle.
- *Magnitude*: Computing the L2-norm gradient magnitude similar to what is used in [6] requires a square root operation which is complex to implement in hardware. An L1-norm magnitude, which doesn't require a square root, is used in this work instead. L1-norm and L2-norm are not linearly dependent but they are strongly correlated. The overall detection accuracy does not degrade with the L1-norm magnitude, which suggests that HOG feature is less sensitive to the gradient magnitude and more sensitive to the gradient orientation.

[2]Different gradient filters are tested in [6] like 1-D, cubic, 3×3 Sobel as well as 2×2 diagonal filters. Simple 1-D [-1 0 1] filter works the best.

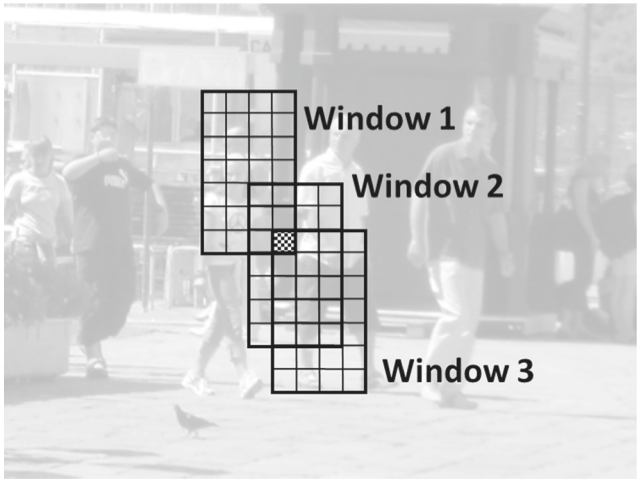

Figure 8 Example of a shared cell across overlapped windows. 8×4 cells per window are shown for simplicity.

As shown in Fig. 2, each cell requires its neighboring 8 cells to create four overlapped blocks for normalization. Accordingly, the 9-bin cell histogram must be stored in a line buffer so that it can be used to compute the normalized histogram with respect to the different blocks. The buffer stores 3 columns of cell histograms. Each histogram bin requires 14-bit, resulting in a total of 126-bit per a cell histogram.

The normalization is then done by dividing the 9-bin histogram by the energy (L2-norm) of each of the four overlapped blocks. Unlike the gradient magnitude, using L1-norm here to compute the block energy results in about 5 % degradation in performance [6]. The L2-norm is computed from sum of square of the histogram bins across the four corresponding cells for each block. The square root is then taken using a simple non-restoring square root module; which is time shared across the four blocks. Finally, 9 sequential fixed point dividers are used (one per bin) to

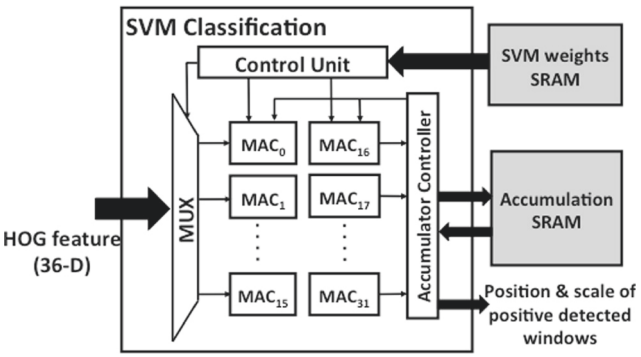

Figure 9 Block diagram of the on-the-fly SVM classification unit.

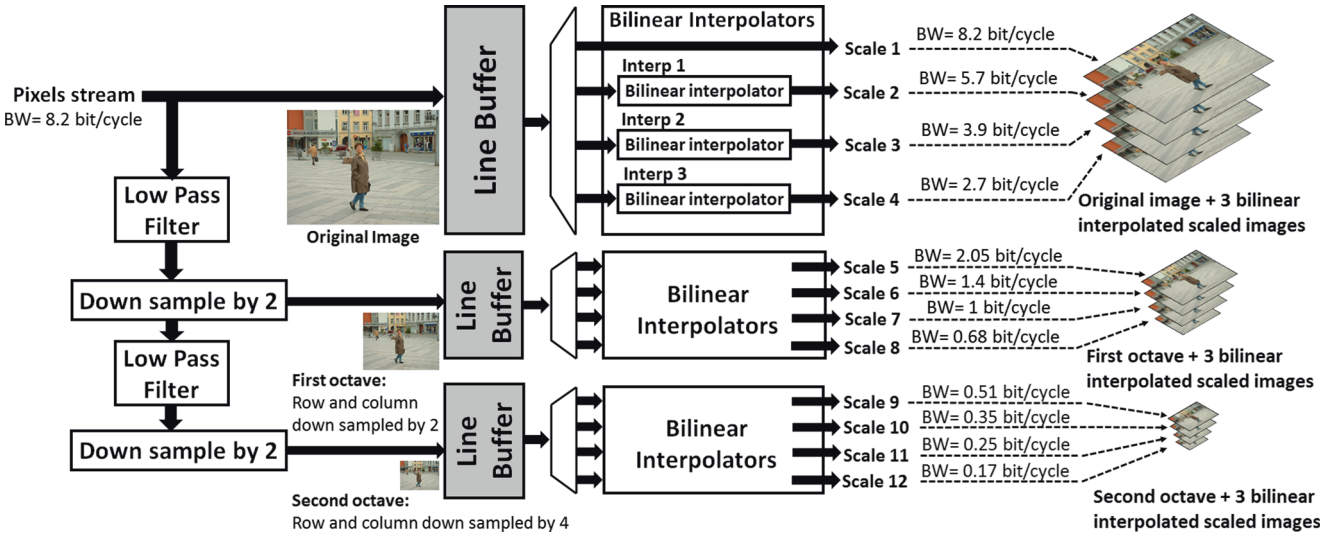

Figure 10 Scale generation architecture with average input and output bandwidths.

generate the normalized HOG feature. These fixed-point dividers are customized to exploit the fact that the normalized values are always fractions (i.e., there is no integer part in the division output). Based on simulations, the bit-width for each normalized bin is chosen to be 9-bit to maintain the detection accuracy. After normalization, the output is a 36-D vector representing the HOG feature.

4.1.2 SVM Classification

In this work, a linear SVM classifier is trained off-line and its weights are loaded to an on-chip buffer, so that the detector can be configured for different objects. The bit-width of the SVM weights is reduced to minimize both size and bandwidth of the on-chip buffer. The 4,608 SVM weights (representing 128×64 pixels per detection window) are quantized to a 4-bit signed fixed-point, with a total memory size of 0.018 Mbit.

An on-the-fly approach similar to [13] is carried out for classification. The HOG feature of each cell is immediately used for classification once it is extracted so that it is never buffered or recomputed. This reduces on-chip memory requirements and external memory bandwidth as each pixel is only read once from the off-chip frame buffer. All calculations that require the HOG feature must be completed before it is thrown away. Figure 8 shows a simple example, with a small detection window size of 8×4 cells, of how several detection windows share a cell. Each cell effectively appears, and must be accumulated, at all positions in overlapped detection windows. For a 128×64 pixels detection window, each cell is shared with (16×8 = 128) windows, but at different positions within each window.

The on-the-fly classification block is shown in Fig. 9. The processing is done using a collection of MACs. Each MAC contains two multipliers and one adder to compute a partial dot product of 2 values of the detection window features and the SVM weights, and another adder to accumulate the partial dot products. Thus, it takes 18 cycles to accumulate all 36 dimensions for a given cell position. Two columns of cells, out of the eight columns in the detection window, are processed in parallel, requiring 16×2 MACs. Accordingly, 18×4 cycles are required to complete the 4,608 multiplications for the whole detection window. Using this approach, the 17-bit accumulation values are stored instead of the HOG features (36×9-bit), resulting in a 19× reduction in the required buffer size.

4.2 Scale Generator

A scale generator is used to generate the image pyramid, as shown in Fig. 3. The key blocks required to generate the pyramid include low pass filters, down samplers, pixel line buffers and interpolators as shown in Fig. 10.

Table 1 Scale factor effect on detection accuracy, with number of cells per 1080HD frame in the image pyramid. AP numbers are calculated on the INRIA person dataset [5].

Scale Factor	AP	Scales	Cells	Increase
Single-scale	0.166	1	32,400	1.0×
2	0.275	4	43,030	1.3×
1.4	0.337	7	65,530	2.0×
1.3	0.372	9	78,660	2.4×
1.2	**0.391**	**12**	**104, 740**	**3.2×**
1.1	0.398	23	184,450	5.7×
1.05	0.401	44	344,220	10.6×

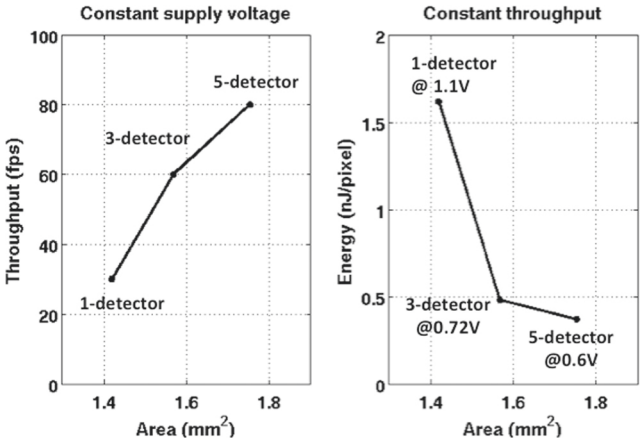

Figure 11 Throughput with constant supply voltage of 0.72 V (*left*) and energy with constant throughput of 1080HD at 60 fps (*right*) for the three different detectors configurations.

4.2.1 Scale Factor Selection

There is a trade-off in selecting the scale factor between the detection accuracy and the number of cells to process. Table 1 shows an exponential increase in the number of cells per frame as more scales are used (i.e., reducing the scale factor). Using a scale factor of 1.05 in the baseline implementation [6] increases the workload by more than 10× compared to a single scale. In this work, a scale factor of 1.2 is chosen as it introduces only 0.01 reduction in AP, with an increase of only 3.2× in the workload. For a 1080HD frame and a scale factor of 1.2, 12 scales per frame are required in the image pyramid. This careful selection of the scale factor results in a 3.3× reduction in the number of cells generated per frame compared to the baseline implementation.

4.2.2 Scale Generation Architecture

Figure 10 shows a block diagram of the scale generator module. Pixels are streamed-in once from the off-chip frame buffer. Low pass filters are used to process the pixels to prevent aliasing before downsampling by two and by four, generating two octaves. The original and the two octaves images are partially stored in on-chip line buffers. The buffers store 25 columns of pixels of each image, which is sufficient to generate the different scales. The buffers are shared across scales within the same octave.

As shown in Fig. 10, the 12 required scales are generated as follows: the fifth and ninth scales, which ideally would have a scaling factor of 2.07 and 4.3 respectively, are approximated to be the octaves (i.e. scale factors of 2 and 4 respectively) to reduce number of calculations. Using bilinear interpolation, three scales are generated from each octave with scale factors of 1.2, 1.44 and 1.73. The bilinear interpolation for the scaled image generation begins as soon as the supporting pixels are available in the shared pixel line buffers. An on-the-fly interpolation is used so that a minimum number of pixels is buffered before interpolation.

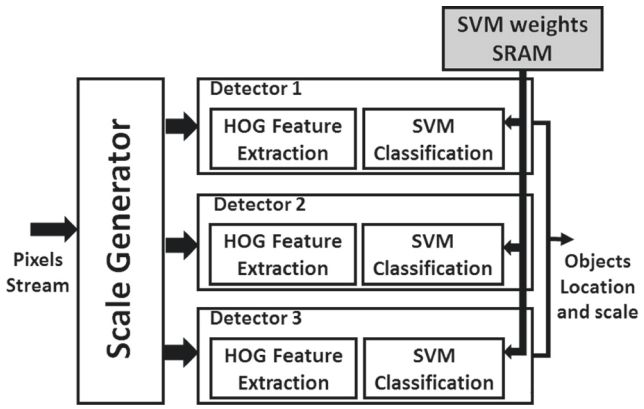

Figure 12 Object detection system architecture.

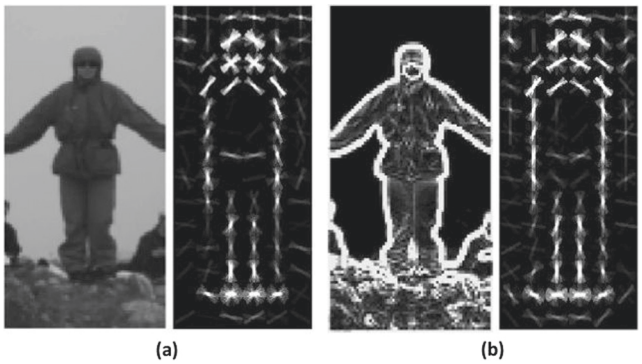

Figure 13 Pedestrian image with the corresponding trained SVM template in (**a**) original and (**b**) gradient magnitude representations. Image is taken from INRIA person dataset [5].

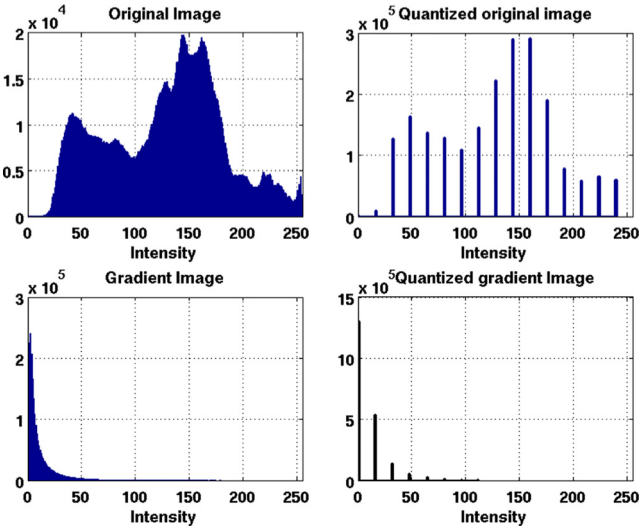

Figure 14 Histogram of pixel intensities. Left column shows 8-bit bit-width. Right column shows coarse quantization.

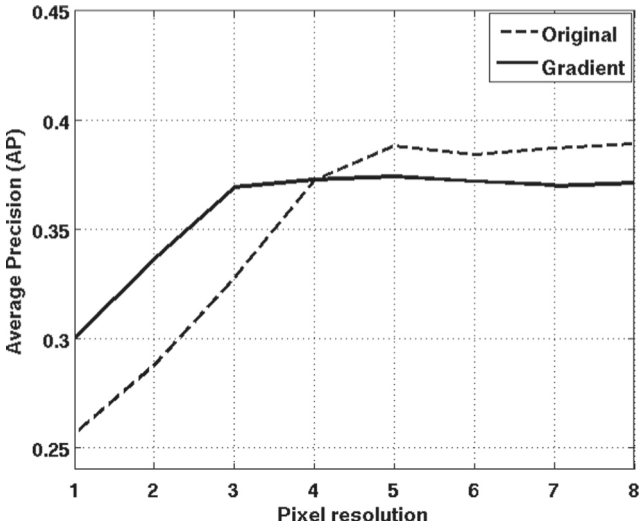

Figure 15 Average precision (AP) values versus pixel resolution for both original and gradient magnitude images.

The generated scales are passed directly to feature extraction module and no scaled images are stored on-chip after interpolation.

5 Parallelism and Voltage Scaling

Multi-scale detection increases the power consumption relative to single-scale due to the scale generation overhead and the processing of the additional scales. If a single detector is used, clock frequency and voltage must be increased to process these additional scales while maintaining the same throughput. Thus in this work, multiple parallel detectors are used in order to reduce the clock frequency and voltage while maintaining the same throughput.

Table 2 Area and power breakdown for object detector architecture for both original and gradient magnitude images. Numbers are based on post-layout simulations with 45nm SOI CMOS process.

		Original	Gradient
Area (kgates)	Pre-proc.	n/a	7
	Scale Gen.	240	167
	Detector 1	90	86
	Detector 2	102	100
	Detector 3	133	130
	Total	**565**	**490**
Power (mW)	Pre-proc.	n/a	0.5
	Scale Gen.	17.10	9.30
	Detector 1	13.10	10.10
	Detector 2	11.60	9.50
	Detector 3	10.15	8.05
	SVM mem.	7.85	7.85
	Total	**59.80**	**45.30**

Three different configurations are tested for the parallel detectors architecture: one, three and five detectors. Figure 11 shows the trade-off between area, power and throughput in each architecture using column raster scan mode.[3] On the left of Fig. 11, the supply voltage is held constant at 0.72V, and a throughput of 30, 60, and 80 fps is achieved by the one, three and five detectors respectively. On the right of Fig. 11, the throughput is held constant at 60 fps, and the supply voltage is varied for each configuration to evaluate the energy consumption. To achieve a constant 60 fps throughput, the supply voltage must be increased to 1.1 V for the one-detector configuration, and can be decreased to 0.6 V for the five-detector configuration. As expected, large energy reduction is achieved by using three parallel detectors compared to a single detector due to voltage scaling. Although the five-detector architecture gives a slightly lower energy point, the three-detector architecture is selected because it offers a better energy versus area trade-off. The same behavior is expected regardless of cell processing order.

A simple replication of the hardware to implement the parallel detectors results in extra area and power consumption, which can be avoided. The parallel detectors are using the same SVM template to detect one object. As a result, a separate buffer for SVM weights in each detector means replicating information and wasting memory area and bandwidth. In this architecture, the detectors are carefully synchronized such that they share the same SVM weights at any moment. This enables using only one buffer for SVM weights for all detectors, which results in 3× reduction in memory size and bandwidth, and 20 % reduction in the overall system power.

Figure 12 shows the overall detection system. The pixels from the 12 scales are distributed to three parallel detectors. The distribution is done such that the three detectors have balanced workloads. The original HD scale is passed to Detector (1), the next 2 scales to Detector (2), and the remaining 9 scales to Detector (3). All three detectors are identical except that the size of the histogram and the accumulator buffers are different based on the number and the size of the scales processed by each detector. The exact memory sizes in each detector are shown in Table 5. With three parallel detectors and voltage scaling, a 3.4× energy saving is achieved compared to a single detector as shown in Fig. 11.

[3]The energy numbers for 0.6 V and 1.1 V supplies are estimated from a ring oscillator voltage versus power and frequency curves. SRAM minimum voltage is 0.72 V.

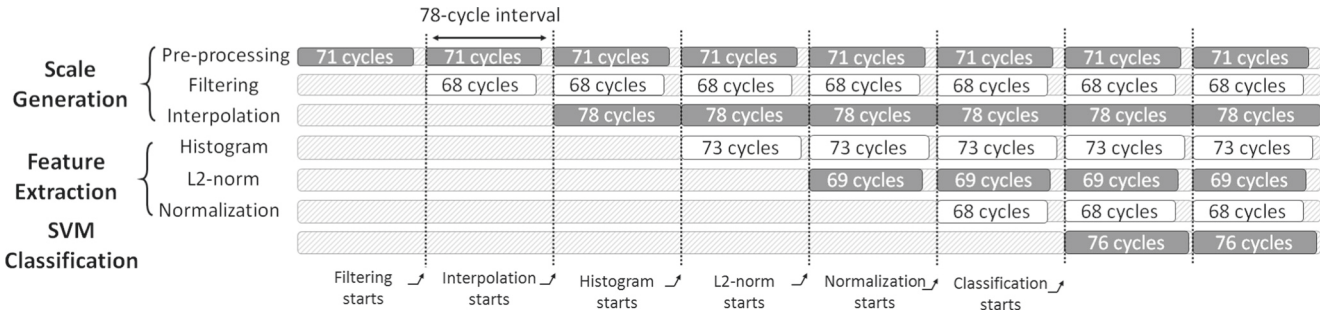

Figure 16 Detection architecture pipeline with number of cycles required by each module.

6 Image Pre-Processing

Image pre-processing can reduce the overhead cost of generating the scales, with minor impact on the detection accuracy.

6.1 Coarse Resolution of Pixel Intensity

The size of the pixel line buffers used in the scale generator block is about half of the overall system memory size. One way to reduce the size of these buffers is to quantize the pixel intensity below the conventional 8-bit. This also reduces the logic of the multipliers in the interpolation block. Figure 15 shows the AP versus the pixel intensity bit-width (dashed line). Because the HOG feature is fairly robust to quantization noise, the intensity can be quantized down to 4-bit with an AP loss of only 0.015. Going from 8-bit to 4-bit pixel intensity results in a 50 % reduction in the pixel line buffers size. However, this can be further reduced as discussed in the next section.

6.2 Detection on Gradient Image

The detection accuracy mainly depends on the features being able to capture the main characteristics of the object. Since the HOG feature is a function of edge orientations, it should have consistent detection performance on other image representations that preserve edge orientations. Figure 13 shows two representations of the same pedestrian: the left is the original intensity image, and the right is the gradient magnitude image. The gradients are calculated using a simple [-1 0 1] filter. Edges that compose the pedestrian contour are visible in both images. To further demonstrate that detection on gradient magnitude images is reasonable, Fig. 13 shows also the trained SVM templates on both original and gradient magnitude training images. Both templates capture similar pedestrian characteristics (e.g. head, shoulders, legs).

Table 3 Optimized fixed point bit-width.

Parameter	Sign	Integer	Fraction
Gradient magnitude	0	8	0
Cell histogram bin	0	14	0
Block energy	0	26	0
HOG feature bin	0	0	9
SVM weights	1	0	3
SVM accumulator	1	3	13

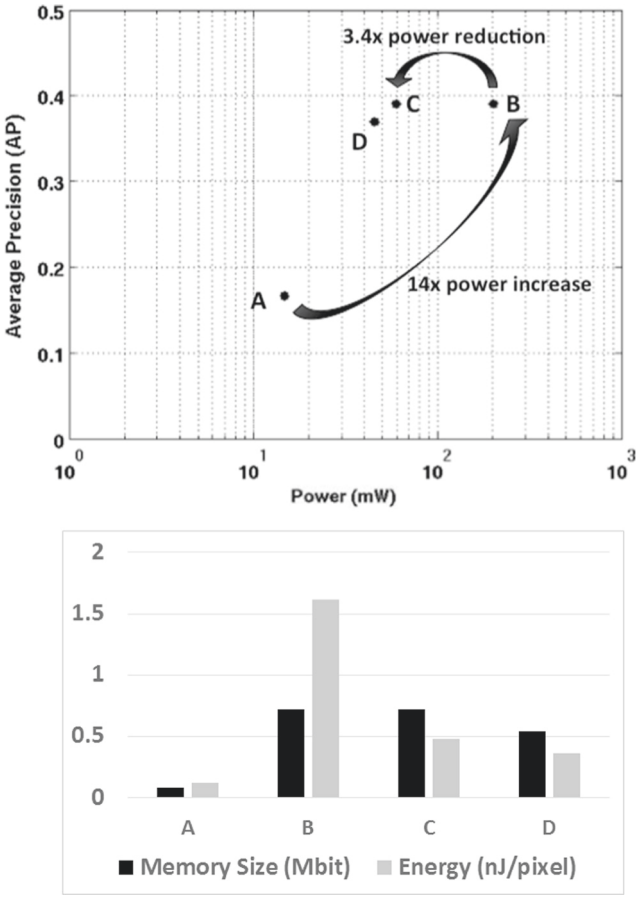

Figure 17 AP, power and memory sizes for different object detection architectures. **a** Single-scale with one detector at 0.6 V. **b** Multi-scale with one detector at 1.1 V. **c** Multi-scale with three parallel detectors at 0.72 V. **d** Multi-scale with three parallel detectors and pre-processing at 0.72 V.

Table 4 Comparison between the memory size in column raster scan and row raster scan architectures. Memory size in (Mbit).

	Column scan	Row scan
Scale Generator	0.136	0.243
Histogram	0.289	0.520
Accumulator	0.095	0.340
SVM Memory	0.018	0.018
Total	**0.538**	**1.121**

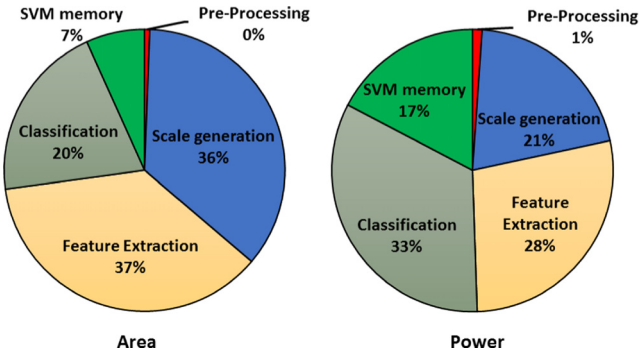

Figure 19 Area and power breakdowns for the overall column scan object detection system.

The motivation behind processing gradient magnitude images is to further reduce the pixel intensity bit-width, and to reduce the switching activity in the hardware because the gradient magnitude image is usually more sparse. Figure 14 shows the intensity histograms for the original and the gradient magnitude representations of an example image. The original image pixels intensities are well distributed across the whole dynamic range of 8-bit. However in the gradient magnitude image, most of the pixels intensities are concentrated around low values and do not cover the whole dynamic range. Thus, the gradient magnitude image can use fewer bits per pixel because of the dynamic range reduction.

Figure 15 shows that the original images give a 0.02 better AP at high pixel intensity bit-width compared to the gradient magnitude images (solid line). At 4-bit, both original and gradient magnitude images have the same AP. Reducing the bit-width to 3-bit in gradient magnitude images approximately maintains the same AP and has a 25 % smaller pixel line buffer size.

The image pre-processing results in a 24 % reduction in the overall system power. The area and power breakdown for both 8-bit original image and 3-bit gradient magnitude image detectors are shown in Table 2. The pre-processing required for the gradient magnitude image detection architecture introduces very small area and power overhead. However, it results in a 45 % power reduction in the scale generator block. The pixel line buffers size is reduced from 0.363 Mbit to 0.136 Mbit. Smaller multipliers are used in the interpolation unit, resulting in 30 % area saving. The detector power is also reduced by 20 % due to smaller subtractors and accumulators in the histogram generation unit, and due to the reduction in switching activity in the data-path.

7 Results

As mentioned in Section 4, this architecture is cell-based where one cell is processed at each stage in the pipeline. Figure 16 shows a timing diagram of the detection pipeline with different top level modules. These modules are running in parallel as shown and each one is processing a different cell at a time in unit time intervals of 78 cycles. Number of cycles is balanced between modules to reduce idle times and maximize throughput. Each module is pipelined internally as well to maximize the operating clock frequency.

7.1 Detection Accuracy

The INRIA person dataset [5] was used to evaluate the impact of the modified parameters on the detection

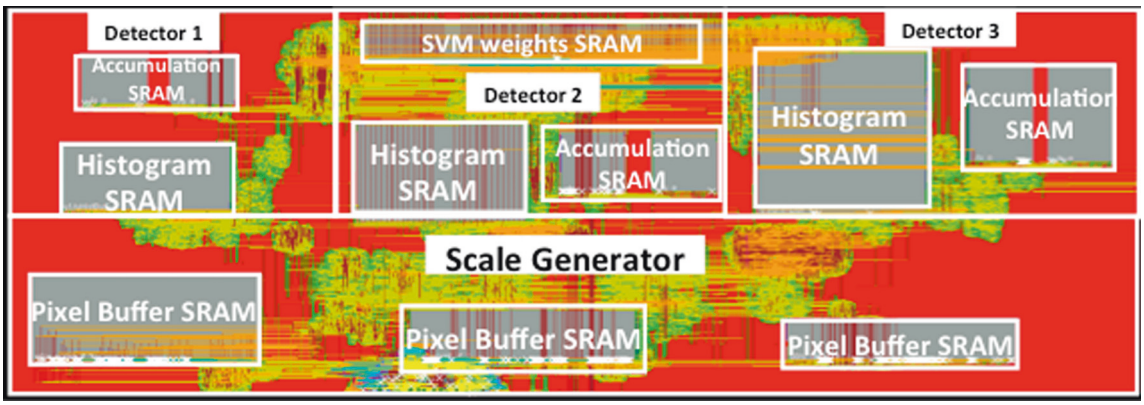

Figure 18 Layout of the object detector core (column scan architecture).

Table 5 The size and the bandwidth of different memories in the design.

Block	Bandwidth (MB/s)		Memory	Size	Dimension
	Read	Write		(Mbit)	address × word
Scale generator	980	255	Pixel buffer 1	0.078	1088×75
			Pixel buffer 2	0.039	544×75
			Pixel buffer 3	0.019	272×75
Histogram buffer	1,009	112	Detector 1	0.055	2048×28
			Detector 2	0.080	3040×28
			Detector 3	0.154	5760×28
Accumulator	1,743	1,743	Detector 1	0.018	544×17(×2)
			Detector 2	0.027	832×17 (×2)
			Detector 3	0.050	1536×17 (×2)
SVM weights	7,776	0		0.018	144×128

accuracy. These modifications include: using L1-norm for the gradient magnitude, fixed-point numbers representation (shown in Table 3), approximating the image pyramid with a scale factor of 1.2, and image pre-processing. Our implementation, which supports multi-scale detection, without pre-processing is close to the original HOG algorithm [6] with 0.389 AP compared to 0.4. With pre-processing, our implementation gives 0.369 AP.

7.2 Architectural and Algorithmic Optimization Results

Figure 17 shows the design space of the detection accuracy, the memory size and the power numbers for different architectures at the same throughput (1080HD video at 60 fps).[4] Our three main contributions can be shown as follows:

1. Introducing multi-scale detection boosts the detection accuracy by **2.4**×. The overhead of the image pyramid generation and the processing of the new scales results in **14**× increase in power and **8.8**× increase in memory size compared to a single detector (A to B in Fig. 17).
2. Parallelism reduces the power by **3.4**× due to voltage and frequency reduction without affecting the detection accuracy. No change in the memory size is achieved (B to C Fig. 17).
3. Image pre-processing reduces multi-scale memory and processing overhead, resulting in a **24%** overall power reduction and a **25 %** overall memory size reduction (C to D in Fig. 17).

[4]Energy numbers for 0.6 V and 1.1 V supplies are estimated from a ring oscillator voltage versus power and frequency curves. SRAM minimum voltage is 0.72 V.

7.3 Image Scanning Order

As discussed in Section 4, two scan modes can be implemented; column and row raster scans. Both modes use similar architecture and layout floorplan. The only difference is the on-chip buffers size and their corresponding address decoder logic. Row line buffers are used in row raster scan architecture and column line buffers are used in column raster scan architecture. Row raster scan results in higher memory size because usually the frame width is larger than its height. For example, the ratio between 1080HD frame width to its height is 16:9; thus a column raster scan would reduce the memory size by approximately $\frac{16}{9}\times$. However, if we account for the fact that cameras typically output pixels in row raster scan order, processing the cells in the same order will reduce latency and additional memory controller complexity to reorder pixels from row to column order. The decision of processing order will depend on the overall system design parameters, such as camera specification.

Table 4 shows a comparison between the two architectures with memory size numbers. Comparing row to column raster scans, the scale generator and the histogram buffers have an increase of 1.8× in their sizes, which is approximately the ratio between the width and the height of a 1080HD frame. The SVM buffer size doesn't change because it stores the same SVM template in both cases. The accumulator buffers on the other hand have double the increase with 3.6×. The difference is that the accumulator buffers change from storing 8 columns of the partial dot product values in the column raster scan architecture to storing 16 rows in the row raster scan architecture. The numbers 8 and 16 are the detection window width and height respectively. The overall on-chip memory size increase from column to row raster scan architectures is about 2×.

Table 6 Comparison between this work (three time-shared parallel detectors, with image pre-processing) and the implementation in [13].

	[13]	This work	
		Column	Row
CMOS Technology	65 nm	45 nm SOI	45 nm SOI
Area (mm^2)	3.3×1.2	2.8×0.96	3.2×1.08
Gate count (kgates)	502	490	498
Memory size (Mbit)	1.220	0.538	1.121
Image resolution	1920x1080	1920x1080	1920x1080
Frequency	42.9 MHz	270 MHz	270 MHz
Frame rate	30 fps	60 fps	60 fps
Scales	1	12	12
Supply	0.7 V	0.72 V	0.72 V
Power (mW)	40.3	45.3	58.5
Energy (nJ/pixel)	0.648	0.364	0.470
AP	0.166	0.369	0.369

7.4 Post-Layout Results

The core layout of the column scan mode architecture is shown in Fig. 18. Area and power breakdown for the main computation blocks of the overall system is shown in Fig. 19. The classification and the SVM buffer consume about 50 % of the system power. The remaining 50 % of the power is divided between the feature extraction and the scale generation. Although the SVM buffer has a relatively smaller size compared to the memories in the scale generation and the feature extraction blocks, it consumes large power due to its high bandwidth. Table 5 shows a breakdown of various memory blocks size and bandwidth. Note that the SVM weights buffer has a zero write bandwidth, assuming that the template weights are loaded only at the beginning of the detection process and never changed.

Table 6 shows a comparison between this work with both scan modes and the ASIC implementation in [13].[5] Both designs can process 1080HD videos. To be able to process 30 fps, the design in [13] has dual cores processing only one scale, resulting in poor detection accuracy. In this work, a 6.4× increase in throughput is required relative to [13]: 3.2× to support multi-scale and 2× to support 60 fps rather than 30 fps in [13]. Although this work supports multi-scale detection, the size of the on-chip memory is only 0.538 Mbit in the column raster scan architecture. Doubling the memory size in the row raster scan architecture still gives a comparable memory size. Comparing column and row raster scan architectures, 30 % increase in power is reported for row raster scan, mainly because of the increase in on-chip memory size.

[5] AP number is not reported in [13]. This number is from single scale HOG detection simulation.

8 Conclusion

Multi-scale support is essential for robust and accurate detection. However, without any architectural optimization, the scale generation and processing would result in a **14**× power consumption increase, which is a concern for energy-constrained applications. An efficient architecture is presented in this work to generate the image pyramid. Parallelism and voltage scaling result in a **3.4**× power reduction. Image pre-processing reduces the scales generation overhead, and results in **24 %** reduction in the overall system power. Two types of image scan modes are presented, column and row raster scans. Using 45 nm SOI CMOS ASIC technology at a supply voltage of 0.72 V, this design can process 1080HD video at 60 fps, with a total power consumption of 45.3 mW and 58.5 mW for column and row raster scan architectures respectively.

Acknowledgments Funding for this research was provided by Texas Instruments and the DARPA YFA grant N66001-14-1-4039. The authors would like to thank Xilinx University Program (XUP) for equipment donation.

References

1. Haltakov, V., Belzner, H., & Ilic, S. (2012). Scene understanding from a moving camera for object detection and free space estimation. In *Proceedings IEEE Intelligent Vehicles Symposium* (pp. 105–110).
2. Dollar, P., Wojek, C., Schiele, B., & Perona, P. (2012). Pedestrian Detection: An Evaluation of the State of the Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(4), 743–761.
3. Meingast, M., Geyer, C., & Sastry, S. (2004). Vision based terrain recovery for landing unmanned aerial vehicles. In *Proceedings IEEE Conference on Decision and Control*, (Vol. 2 pp. 1670–1675).
4. Myers, B., Burns, J., & Ratell, J. (2001). Embedded Electronics in Electro-Mechanical Systems for Automotive Applications. *SAE Technical Paper* (2001-01-0691).
5. INRIA Person Dataset for Pedestrian Detection. http://pascal.inrialpes.fr/data/human/.
6. Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Vol. 1 pp. 886–893).
7. Dollar, P., Belongie, S., & Perona, P. (2010). The Fastest Pedestrian Detector in the West. In *Proceedings of the British Machine Vision Conference* (pp. 68.1–68.11).
8. Jinwook, O., Gyeonghoon, K., Injoon, H., Junyoung, P., Seungjin, L., Joo-Young, K., Jeong-Ho, W., & Hoi-Jun, Y. (2012). Low-Power, Real-Time Object-Recognition Processors for Mobile Vision Systems. *IEEE Micro*, *32*(6), 38–50.

9. Wei, Z., Zelinsky, G., & Samaras, D. (2007). Real-time Accurate Object Detection using Multiple Resolutions. In *Proceedings IEEE International Conference on Computer Vision* (pp. 1–8).
10. Benenson, R., Mathias, M., Timofte, R., & Van Gool, L. (2012). Pedestrian detection at 100 frames per second. In. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2903–2910).
11. Bauer, S., Kohler, S., Doll, K., & Brunsmann, U. (2010). FPGA-GPU architecture for kernel SVM pedestrian detection. In. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (pp. 61–68).
12. Mizuno, K., Terachi, Y., Takagi, K., Izumi, S., Kawaguchi, H., & Yoshimoto, M. (197). Architectural Study of HOG Feature Extraction Processor for Real-Time Object Detection. In *Proceedings IEEE Workshop on Signal Processing Systems*.
13. Takagi, K., Mizuno, K., Izumi, S., Kawaguchi, H., & Yoshimoto, M. (2013). A sub-100-milliwatt dual-core HOG accelerator VLSI for real-time multiple object detection. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 2533–2537).
14. Hahnle, M., Saxen, F., Hisung, M., Brunsmann, U., & Doll, K. (2013). FPGA-Based Real-Time Pedestrian Detection on High-Resolution Images. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 629–635).
15. Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Journal of Machine Learning*, *20*(3), 273–297.
16. Cao, T., & Deng, G. (2008). Real-Time Vision-Based Stop Sign Detection System on FPGA. In *Proceedings Digital Image Computing: Techniques and Applications* (pp. 465–471).

Amr Suleiman received the B.S. in Electronics and Electrical Communications Engineering in 2008 from Cairo University, Egypt, and the M.S. degree in Electrical Engineering from MIT in 2013. He is currently working toward his Ph.D. under the supervision of Prof. Vivienne Sze. His work focuses on developing new energyefficient implementations for machine vision algorithms (e.g. detection, recognition, and tracking). In addition to optimizations at the architecture and circuit level, system optimization is done at the algorithmic level to efficiently trade-off accuracy and complexity.

Vivienne Sze received the B.A.Sc. (Hons) degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2004, and the S.M. and Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2006 and 2010 respectively. In 2011, she received the Jin-Au Kong Outstanding Doctoral Thesis Prize in Electrical Engineering at MIT.

She has been an Assistant Professor at MIT in the Electrical Engineering and Computer Science Department since August 2013. Her research interests include energy-aware signal processing algorithms, and low-power circuit and system design for portable multimedia applications. Prior to joining MIT, she was a Member of Technical Staff in the Systems and Applications R&D Center at Texas Instruments (TI), Dallas, TX, where she designed low-power algorithms and architectures for video coding. She also represented TI at the international JCT-VC standardization body developing HEVC. Within the committee, she was the primary coordinator of the core experiment on coefficient scanning and coding, and has chaired/vice-chaired several ad hoc groups on entropy coding. She is a co-editor of "High Efficiency Video Coding (HEVC): Algorithms and Architectures" (Springer, 2014).

Prof. Sze is a recipient of the 2014 DARPA Young Faculty Award, 2007 DAC/ISSCC Student Design Contest Award and a co-recipient of the 2008 A-SSCC Outstanding Design Award. She received the Natural Sciences and Engineering Research Council of Canada (NSERC) Julie Payette fellowship in 2004, the NSERC Postgraduate Scholarships in 2005 and 2007, and the Texas Instruments Graduate Woman's Fellowship for Leadership in Microelectronics in 2008.