A LOW POWER HARDWARE IMPLEMENTATION OF MULTI-OBJECT DPM DETECTOR FOR AUTONOMOUS DRIVING

Alaa Ali *Oladiran G. Olaleye* *Bappaditya Dey* *Kasem Khalil* *Magdy A. Bayoumi*

The Center for Advanced Computer Studies (CACS)
University of Louisiana at Lafayette, LA, USA
{ama8030, ogo8842, bxd9836, kmk8148, mab0778}@louisiana.edu

ABSTRACT

Object detection is a fundamental process in traffic management systems and self-driving cars. Deformable part model (DPM) is a popular and competitive detector for its high precision. This paper presents a programmable, low power hardware implementation of DPM based object detection for real-time applications. Our approach employs a very fast object detection pipeline with complementary techniques such as fast feature pyramid, Fast Fourier Transform (FFT) and early classification to accelerate DPM with a reasonable accuracy loss and achieves a speed-up of 50x and 6x over original DPM and cascade DPM respectively on single core CPU. The hardware circuit uses 65nm CMOS technology and consumes only 36.5mW (0.81 nJ/pixel) based on the post-layout simulation. The ASIC has an area of 3362 kgates and 295.5 KB on-chip memory and the design utilizes two simultaneous engines to process two independent object categories with 8 deformable parts per category.

Index Terms— Object Detection, DPM, FFT, HOG, Autonomous Driving

1. INTRODUCTION

The main challenges of autonomous cars relate to two dynamic problems. First, a detailed investigation of the roads and terrains it is passing by to acquire knowledge about the dynamic environment (free-way, neighborhood). Second, an accurate detection and tracking of differentiable objects such as pedestrians, road-side light-post, traffic signals, etc. In addition to that, self-driving cars should be capable enough to derive most accurate inference about its driving context (perception capability to judge the immediate intention of other drivers or a pedestrian waiting at an intersection to cross-walk). Autonomous vehicles include grand challenges to understand physical world [1, 2] and achieve transportation capabilities of a classic car. Current autonomous driving techniques depends mainly on computer vision besides GPS, RADAR and LIDAR. Object detection and accurate vehicle detection are essential problems for practical intelligent applications, such as autonomous vehicles, smart traffic and driver assistance.

The deformable part model (DPM) [3] is very popular and challenging object detector for being a mature and stable technique. DPM learns a multi-component mixture model based on Histogram of Oriented Gradients (HOG) [4] and has a merit of large appearance variations handling for challenging benchmarks. However, original DPM takes more than 10 seconds (single thread) per image on Pascal VOC [5].

Multiple accelerated techniques are proposed to speed up DPM while keeping close to its high detection rates such as CascadeDPM [6], CoarseDPM [7], FFT-DPM [8], Rapid-DPM [9], FastDPM [10] and R-DPM [11].

We propose a low power hardware implementation of DPM object detection pipeline based on three complementary components to speedup DPM. We provide an extensive performance evaluation in terms of time and accuracy on detection benchmarks.

The remainder of the paper is organized as follows: Section 2 surveys briefly the related work with focus on DPM. Section 3 presents details of our proposed framework. Section 4 shows experimental results and performance evaluation on benchmark datasets. Finally section 5 concludes the paper.

2. RELATED WORK

DPMs [3] have the potential to have more competitive runtimes to squeeze the involved extensive computations and bottlenecks. Recent research work and multiple approaches have been introduced to speed up DPMs technique. Felzenszwalb et al. proposed CascadeDPM [6], which eliminates unpromising hypotheses efficiently and achieves about $14\times$ processing time speed-up on PASCAL 2007 dataset [5] with a reasonable precision loss in comparison to original DPM [3]. Pedersoli et al. proposed a hierarchical part based model with coarse-to-fine procedure, CoarseDPM [7], to prune hypotheses at low cost. Iasonas uses Dual-Tree Branch-and-Bound (DTBB) [9] to calculate the cost function's upper bounds of part-based DPM model and result in efficient acceleration with almost similar detections.

Recently, Yan et al. proposed FastDPM [10] to improve time performance through alternative strategies of Star-Cascade technique [6] based on a lookup table Histogram of Oriented Gradients (HOG) [4] and neighborhood aware cascade. Fast DP-DPM uses deep learning to improve the performance of DPM [12]. However, it is still slow for real-time applications.

In terms of hardware architecture, Kenta et al. presented a simplified HOG strategy oriented for a real-time multi-object detection VLSI processor [13], a dual core architecture is designed to process HDTV video at 30 HZ and fabricated 65 nm chip as a scalable architecture under low power conditions [14]. Amr et al. presented an energy efficient multi-scale HOG architecture [15] on a 45 nm chip to process 1080HD video at 60 HZ consuming 45.3 mW and a programmable energy efficient DPM hardware architecture [16] using 65 nm chip to process HDTV video at 30 HZ and consuming 58.6 mW. These architectures could be utilized in fault prediction and self-healing hardware systems [17].

3. PROPOSED OBJECT DETECTION SYSTEM

Our work is complementary to recent approaches for faster DPMs as we focus on a combination to improve the speed of pyramid construction and classification step, we couple fast feature pyramid technique [18] and LUT HOG features [4, 10] with an optimized FFT classification scheme and then we provide SIMD optimized and multi-threaded version for better speedup results while maintaining accuracy [19].

3.1. Proposed Hardware Architecture

The overall architecture of the proposed DPM hardware detector is composed of the following units: fast feature pyramid generation, Fast Fourier Transform (FFT) unit, SVM classification and SVM early detection and rejection unit. The required units will provide on-the-fly data processing, therefore all calculations are executed as soon as needed data becomes available, which reduces the on-chip memory needed. The proposed detector architecture is a standalone hardware to accelerate detection process, it reads an input image and generates the automatic detection locations. We provide and optimized implementation in order to ignore the need of external storage and reduce the power consumption of whole system.

3.2. Fast Feature Pyramid

We can represent any feature α as a weighted sum of the channel as:

$$\alpha_{\Omega}(I) = \sum_{ijk} \omega_{ijk} C(i,j,k) \tag{1}$$

where I is any image, $C = \Omega(I)$ and $\Omega(I)$ is a low-level shift variant function applying on the image to create the channel

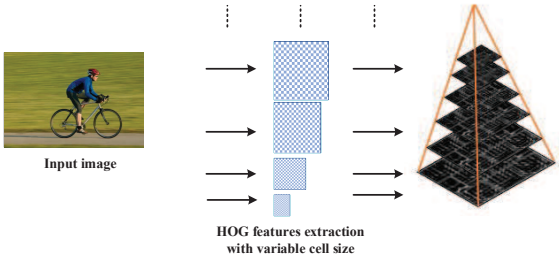

Fig. 1. Generating feature pyramid.

C. Also we can rewrite $\alpha_{\Omega}(I_s)$ as follows for simplicity:

$$\alpha_{\Omega}(I_s) \equiv \frac{1}{h_s \omega_s k} \sum_{ijk} \omega_{ijk} C_s(i,j,k) \tag{2}$$

where $(h_s \times \omega_s)$ is dimensionality of the image I_s sampled at any scale s and k is the layer [18]. Now, if we decompose our initial image I into M smaller images such that $I = [I^1, I^2, ..., I^M]$. We can rewrite Eq. 2 as follows:

$$\alpha_{\Omega}(I) \approx \frac{\sum \alpha_{\Omega}(I^m)}{M} \tag{3}$$

If we can express $\alpha_{\Omega}(I) \approx E[\alpha_{\Omega}(I^m)]$ by considering $[I^1, I^2, ..., I^M]$ as a small image ensemble, where $E[\cdot]$ denotes the expectation over that ensemble of the image I. Then according to Ruderman and Bialeks finding with more details in [18] about functional representation of the statistics of any natural image I, in terms of the sampling scale s at which image ensemble was captured, we can substitute the power law by:

$$\alpha_{\Omega}(I_{s_1}) / \alpha_{\Omega}(I_{s_2}) = (s_1/s_2)^{\lambda_{\Omega}} + \eta \tag{4}$$

where the variable η denotes a measure of deviation from Eq. 4 for any given image. The above computation is significant subject to its dependency on the ratio of the scales at which image ensembles were captured (i.e. s_1/s_2) instead of the corresponding individual ones.

3.3. Pyramid Generation

The HOG-based detection is simple to understand, it's a "'global"' feature to describe an object rather than a collection of local features [4] where the entire object is represented as a feature vector. To compute HOG feature descriptors, an input image is divided into non-overlapping 8×8 pixels size cells to be processed within detection windows. The cells are arranged into overlapping blocks. Orientations histogram is computed for each cell and then normalized with respect to neighbors [4]. A scale generator is utilized to create the image pyramid with multiple scales as shown in Fig. 1.

3.4. Multi-scale Generator Architecture

A block diagram of the scale generator module in Fig. 2. In order to generate multiple scales, input pixels are streamed

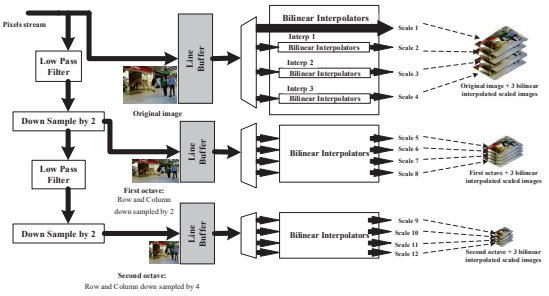

Fig. 2. Scale generator architecture.

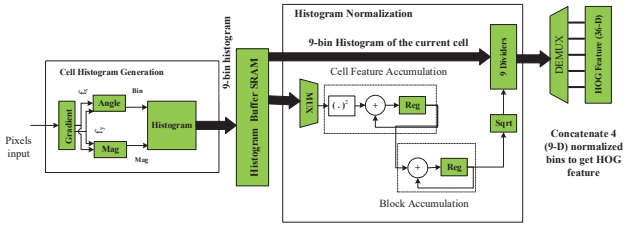

Fig. 3. Block diagram of HOG feature extraction.

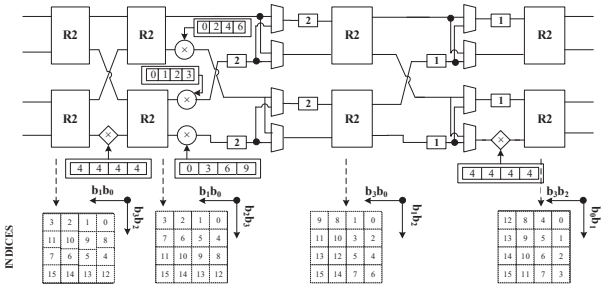

Fig. 4. 4-parallel radix-2^2 feedforward architecture for the computation of the 16-point FFT.

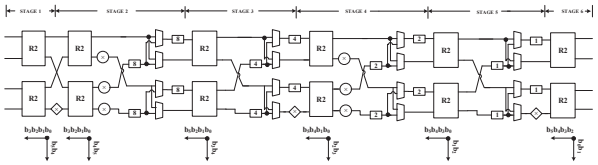

Fig. 5. 4-parallel radix-2^2 feedforward 64-point DIT FFT architecture.

into module from the frame buffer. We use low pass filters to process input pixels before down-sampling at different levels in order to prevent aliasing and then create two separate octaves. All generated images are stored into line buffers partially, we generate mainly the exact scales which are used in further steps to extract HOG features for those scales. Our line buffers will store 30 columns of pixels of each rescaled image, which is sufficient to generate the different scales. These line buffers are shared across the multiple scales created within same octave [15].

3.5. Fast Fourier Transform (FFT)

FFT is used to efficiently compute the discrete Fourier transform (DFT) and it is one of the most important techniques in signal processing. Individual part detectors of part based models are responsible for extracting low-level features from every scale of an image pyramid. The score of a filter evaluated on an image can be computed as:

$$s_{pq} = \sum_{f=0}^{F-1} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} a^f_{p+m,q+n} b^f_{mn} \tag{5}$$

Now, we can compute the responses of a filter considering all possible locations as: $S = \sum_{f=0}^{F-1} a^f \overline{b}^f$ where $\overline{b}$ is the reversed filter. With the above image and filter size, the cost of standard convolution, of complexity $O(LWMN)$, will be replaced with $O(LWlogLW)$ when applying FFT.

We use the Fourier transform to speed-up the different evaluations required of a linear classifier in a multi-scale detection framework. The standard process without FFT is to convolve all HOG features and HOG filters and then to sum all the resulting per-feature scores. Leveraging FFT in our pipeline is inspired from [8], convolutions are done by computing FTs of features pyramid, multiplying them in Fourier domain, and then we compute inverse FT of the result. The convolution process is accelerated by an order of magnitude at least. If we need to convolve L HOG filters and sum them across K HOG features, then the total cost per image is

$$C_{fourier/img} = \underbrace{KC_{FFT}}_{forward\ FFTs} + \underbrace{LC_{FFT}}_{inverse\ FFTs} + \underbrace{KLC_{mul}}_{multiplications} \tag{6}$$

3.5.1. *Radix-2^2 FFT Architecture*

DFT of an input sequence for N point is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, k = 0, 1, ..., N-1 \tag{7}$$

where W_N^{nk} is given by $W_N^{nk} = e^{-j(2\pi/N)nk}$

The FFT is based on the Cooley Tukey algorithm which is widely used in efficient DFT computations when N is a power of two [20]. The complexity of Cooley-Tukey algorithm is O(N log_2 N) [21]. Butterflies and rotations are computed within each stage of the graphs, where the lower edges of the butterflies are always multiplied by -1 for graph simplification.

The flow graph of a radix-2^2 DIF FFT can be acquired from the graph of a radix-2 DIF.Therefore, each stage is broken down: at odd stages, the breaking is done into trivial and non trivial rotation, $\bar{\phi}$ and $\bar{\phi} = \phi$ mod $N/4$ and moving $\bar{\phi}$ to

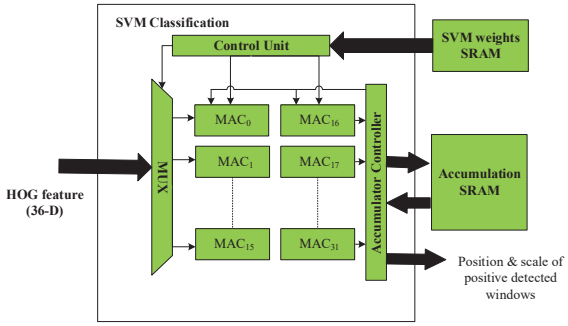

Fig. 6. SVM Classification.

the next stage. Fig. 4 shows a 16-point 4-parallel radix-2^2 butterflies based feedforward FFT architecture.

3.6. Early SVM Classification

After offline training of a linear SVM classifier for the required DPM model in root and parts filters, the model coefficients are loaded to an on-chip buffers. Our detector can be configured for different object categories. For example, in case of INRIA Person benchmark, the $4,608$ SVM weights which represents 128×64 pixels for a pedestrian root filter are quantized into 4-bit fixed-point and a memory size equals to 0.018 Mbit [15]. The SVM classification processing is performed using a number of MACs where each MAC has an adder and two multipliers used to calculate the partial dot product for values of the extracted features per window and SVM coefficients plus an extra adder to accumulate the partial dot products. The classification block diagram is shown in Fig. 6.

The technique for early classification is inspired from a simplified HOG algorithm proposed in [14]. In our work, we use a similar technique to early detect and reject the DPM part candidates in each component (we use 8 parts per component), instead of waiting to calculate all parts score, we compare partial parts score obtained subsequently with early classification threshold and then posterior computations can be skipped.

Table 1. Comparison with ODROID-XU3 board processing Full HD 1920×1080 frames

Platform	Cortex-A7 [16]		Cortex-A15 [16]		DPM [16]		Our Work	
	1 core	4 cores	1 core	4 cores	0.77V	1.11 V	0.77V	1.11 V
Process Technology	28nm HKMG		28nm HKMG		65nm CMOS		65nm CMOS	
Throughput(fps)	0.04	0.10	0.11	0.24	30	60	42	74
Power(mW)	155.6	383.5	1,703.8	3,575.6	58.6	216.5	36.5	182.4
Energy(nJ/pixel)	1,881.8	1,849.0	7,301.2	7,165.7	0.94	1.74	0.81	1.48

4. EXPERIMENTAL RESULTS

Table 2 shows a comparison between our hardware architecture versus other previous hardware object detector based on HOG. Both circuits in [14] and [16] can process full HD frames at 30 fps with lower accuracy than the original DPM algorithm. Our proposed hardware circuit can provide fast multi-scale feature map generation and the whole detection process to run on-chip with utilizing deformable parts in detection, which provide higher the detection rates and robust detection. In the meanwhile, it consumes 34% less energy in comparison to previous architecture [14].

Table 2. Performance comparison on PASCAL VOC 2007 dataset for multiple hardware implementations.

	HOG [14]	DPM [16]	Our Work
Process	65nm	65nm	65nm
Chip size	$4.2 \times 2.1\ mm^2$	$4.0 \times 4.0\ mm^2$	$4.2 \times 4.0\ mm^2$
Input resolution	1920×1080	1920×1080	1920×1080
Multi-scale	one scale	12 scales	12 scales
Deformable parts	No	8	8
Object classes	2	2	2
Frame rate	30	30	42
Frequency	84.3 MHz	62.5 MHz	62.5 MHz
Power	84 mW	58.6 mW	36.5 mW
Energy/pixel	1.35 nJ	0.94 nJ	0.81 nJ
Mean AP	18.5%	26%	31.4%

A comparison between the proposed hardware circuit and a software implementation of DPM algorithm running on ODROID-XU3 board [16] is shown in Table 1. The required pre-processing steps and post processing step of non-maximal suppression for the DPM based detection hardware architecture have a small power consumption and area overhead in comparison to the main intensive computations steps.

The proposed ASIC architecture has less energy consumption than the Samsung embedded processor in about 3 orders of magnitude. These results show performance evaluation on HD frames and it shows that a maximum throughput of 0.24 frames per seconds can be achieved on ODROID-XU3 board with the Samsung processor when using the 4 cores of Cortex-A16.

5. CONCLUSION

In this paper, a novel pipeline is proposed to accelerate deformable part models and achieve a real-time object detection, while keeping the similar high detection rates of DPM. We present a low power and real-time hardware implementation for a DPM based object detector. Our circuit uses a 65 nm CMOS technology and processes 1920×1080 videos at 36 fps while consuming only 36.5 mW and resulting in an energy efficiency of 0.81 nJ/pixel. Experimental results on different benchmarks show that our work is generic and effective in the context of vehicle detection and other object categories.This work presents DPM to real-time applications such as driver assistance and autonomous vehicles and provides object detection to be low power and energy efficient as video compression.

6. REFERENCES

[1] E. Guizzo, "How googles self-driving car works," in *IEEE Spectrum*, 2011.

[2] Hyunggi Cho, Young-Woo Seo, B.V.K. Vijaya Kumar, and R.R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *ICRA*, 2014, pp. 1836–1843.

[3] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *PAMI*, vol. 32, no. 9, pp. 1627–1645, Sept 2010.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, vol. 1, pp. 886–893.

[5] M. Everingham, L. Van Gool, C.I. Williams, J Winn, and A. Zisserman, "The pascal visual object classes challenge 2007 (voc2007) results," *Technical Report*, 2007.

[6] P.F. Felzenszwalb, R.B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *CVPR*, 2010, pp. 2241–2248.

[7] M. Pedersoli, A. Vedaldi, and J. Gonzalez, "A coarse-to-fine approach for fast deformable object detection," in *CVPR*, 2011, pp. 1353–1360.

[8] Charles Dubout and Franois Fleuret, "Exact acceleration of linear object detectors," in *ECCV*. 2012, vol. 7574, pp. 301–311, Springer.

[9] Iasonas Kokkinos, "Bounding part scores for rapid detection with deformable part models," in *ECCV*, 2012, vol. 7585, pp. 41–50.

[10] Junjie Yan, Zhen Lei, Longyin Wen, and S.Z. Li, "The fastest deformable part model for object detection," in *CVPR*, 2014, pp. 2497–2504.

[11] A. Ali, O. G. Olaleye, and M. Bayoumi, "Fast region-based dpm object detection for autonomous vehicles," in *MWSCAS*, 2016.

[12] A. Ali, O. G. Olaleye, B. Dey, and M. Bayoumi, "Fast deep pyramid dpm object detection with region proposal networks," in *ISSPIT*, 2017.

[13] K. Takagi, K. Mizuno, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A sub-100-milliwatt dual-core hog accelerator vlsi for real-time multiple object detection," in *ICASSP*, 2013, pp. 2533–2537.

[14] Kenta Takagi, Kotaro Tanaka, Shintaro Izumi, Hiroshi Kawaguchi, and Masahiko Yoshimoto, "A real-time scalable object detection system using low-power hog accelerator vlsi," *Journal of Signal Processing Systems (JSPS)*, vol. 76, no. 3, pp. 261–274, 2014.

[15] Amr Suleiman and Vivienne Sze, "An energy-efficient hardware implementation of hog-based object detection at 1080hd 60 fps with multi-scale support," *Journal of Signal Processing Systems (JSPS)*, vol. 84, no. 3, pp. 325–337, 2016.

[16] Amr Suleiman, Zhengdong Zhang, and Vivienne Sze, "A 58.6 mw 30 frames/s real-time programmable multi-object detection accelerator with deformable parts models on full hd 1920x1080 videos," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 52, no. 3, pp. 844–855, 2017.

[17] K. Khalil, O. Eldash, and M. Bayoumi, "Self-healing router architecture for reliable network-on-chips," in *ICECS*, 2017.

[18] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *PAMI*, vol. 36, no. 8, pp. 1532–1545, 2014.

[19] A. Ali and M. A. Bayoumi, "Towards real-time dpm object detector for driver assistance," in *ICIP*, 2016, pp. 3842–3846.

[20] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson, "Pipelined radix- 2k feedforward fft architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 23–32, 2013.

[21] James W Cooley and John W Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.