Marek KRAFT, Michał OLEJNICZAK, Michał FULARZ
POZNAN UNIVERSITY OF TECHNOLOGY, FACULTY OF ELECTRICAL ENGINEERING,
INSTITUTE OF CONTROL AND INFORMATION ENGINEERING
3a Piotrowo St., 60-965 Poznan, Poland

A flexible, high performance hardware implementation of the simplified histogram of oriented gradients descriptor

Abstract
In this paper, a high performance, configurable, compact hardware architecture for computing the histogram of oriented gradients (HoG) descriptors is presented. The descriptor computation algorithm is simplified w.r.t. to the original solution, enabling hardware resource cost reduction with only a small accuracy penalty. The proposed architecture can be accommodated to different block sizes and different block grid configurations, enabling its use in a wide range of object detection and recognition tasks with varying region of interest sizes. The resulting architecture is systolic and massively parallel, enabling high throughput processing.

Keywords: HoG, FPGA, image processing.

1. Introduction

Object detection in images and videos is a field of active research. A canonical approach to tackling the problem is the use of an image region descriptor paired with a classifier. One of the most referenced, known and applied methods for image region feature extraction was described in the seminal paper by Dalal and Triggs [1]. The method uses the Histogram of Oriented Gradients (HoG) descriptor to generate a representation that captures the characteristics of the image patch in the detection window. The computation of such features is considered a computationally intensive task, especially if it is to be performed on whole images. As such, the functionality of HoG descriptor computation is a good candidate for implementation as dedicated programmable hardware. Prior works on this subject include [2], in which a highly accurate, floating point implementation of HoG-based classification system was presented. According to the authors, using floating-point operators simplifies the conceptual design, but on the other hand increases the consumption of resources. An ASIC-targeted design with clever simplifications was presented in [3]. A complex, complete design for multi-scale pedestrian detection with HoG and SVM classifier was presented in [4].

In this paper, we present a dedicated hardware architecture that enables the computation of the unnormalized HoG descriptors. The architecture accepts a region of interest of the image that has been constrained beforehand using e.g. a movement detection algorithm or a coarse detector. It is flexible and configurable in terms of detection window grid size and proportions, which enables easy tailoring to a wide range of applications and detected objects, not just pedestrians. Moreover, the simplifications introduced to the original algorithm make the implemented architecture compact.

2. The HoG descriptor

The working principle of HoG is based on counting occurrences of gradient orientations in localized portions of an image. Characteristics and shape of an object within an image are classified by the distribution of the intensity gradients or edge directions. The image is split into small-scale parts called cells and later for each of them a histogram of orientation gradients is calculated.

The descriptor is then the concatenation of these histograms. To increase accuracy of the classifier using local histograms they are contrast-normalized to a unit-length vector within each block. It results in better invariance to changes in illumination and shadowing.

The HOG descriptor provides some key features that other descriptors do not have. Since it works on local cells, it is invariant to photometric and geometric transformations, apart from object orientation. Changes like that would only appear in larger spatial regions. Furthermore, Dalal and Triggs determined that coarse spatial sampling, fine orientation sampling, and strong local photometric normalization gives opportunity for further improvement. The general presentation of HoG descriptor computation is given in Fig. 1.

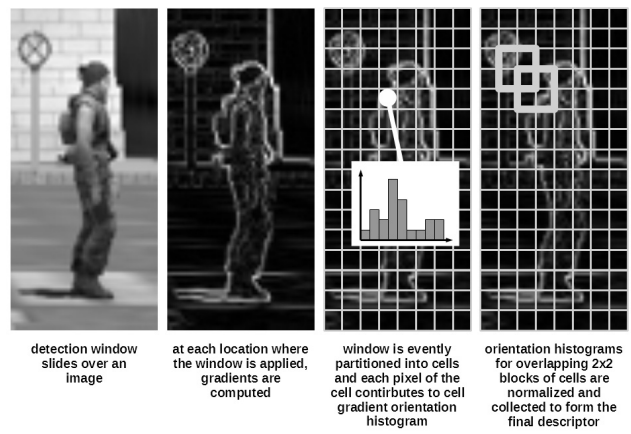


Fig. 1. HoG image region descriptor formation

The first step in computation of many feature detectors is preprocessing aimed at ensuring normalized color and gamma values. However Dalal and Triggs point out that this step can be omitted in HOG descriptor, as the ensuing descriptor normalization achieves the same result.

Typical course of action is to apply the 1-D centered, point discrete derivative mask in one or both directions. Specifically, this method demands filtering the color or intensity data of the image with the following filter kernels:

$$[-1\ 0\ 1], \quad [-1\ 0\ 1]^T$$

The second step of computing HOG is creating the cell histograms. Every pixel inside the cell casts a weighted vote in the orientation histogram based on the values found in the gradient calculation. Shape of the cells can be either rectangular or radial, and the histogram bins are evenly spread over 0 to 180 degrees or 0 to 360 degrees. It depends on whether or not the orientation of the gradient is taken into account. Dalal and Triggs discovered, that unsigned gradients performed best in their human detection experiment when used in conjunction with 9 histogram bins.

Regarding the vote weight, we can represent pixel contribution as a gradient magnitude itself, or some function of the magnitude. Interestingly enough, in tests it was gradient magnitude itself that generally produced the best results. Alternatively the vote weight could incorporate square root of the gradient magnitude, its square or some clipped version of the magnitude itself [1].

To compensate for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger blocks. The concatenated vector of the elements of the normalized cell histograms from every block region is what we call the HOG descriptor. These blocks generally overlap, meaning that each cell commits more than once to the final descriptor. We can distinguish two main

block geometries: rectangular R-HOG blocks and circular C-HOG blocks.

R-HOG blocks are roughly square grids. They are described by three parameters (numbers): channels per block, pixels per cell, and channels per cell histogram. Dalal and Triggs in their experiment discovered that the optimal parameters were four 8×8 pixel cell per block (16×16 pixels per block).

Furthermore, they determined that applying a Gaussian spatial window filtering within each block before tabulating histograms votes in order to weigh pixels around the edge of the block could result in the minor improvement in the performance. The R-HOG blocks turn out to be quite similar to the scale-invariant feature transform (SIFT) descriptors. However, despite their comparable formation, R-HOG blocks are calculated in dense grids at some single scale without orientation alignment, whereas SIFT descriptors are normally computed at sparse, scale-invariant key image points and are rotated to align with dominant orientation of the feature. Furthermore, while SIFT descriptors are used, the R-HOG block are used in conjunction to encode spatial information in a standalone form.

3. Hardware architecture

Schematic representation of the processing pipeline for a single cell is presented in Fig. 2.

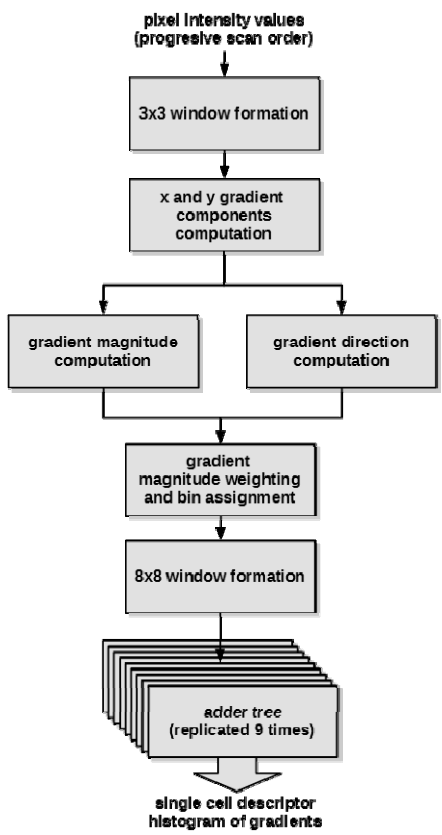


Fig. 2. Block diagram of the processing pipeline computing the descriptor for a single cell

It is assumed that the data (256×256 pixel window) is fed to the processing system in the progressive scan order. The processing pipeline is organized in a way, that enables systolic, parallel processing with a sliding window approach where necessary. Writing a single pixel on the input makes the whole processed region slide by one pixel to the right, and then in a line by line order.

First, the incoming pixels are organized as a 3×3 window which simultaneous access to all pixel values stored in it. This block is composed of internal dual-port RAM memories and register files. This enables the computation of the horizontal and vertical gradient as given by equation (1).

In the next step, the gradient module and direction are computed. The gradient computation module uses an approximation suggested in [5] given by equation (2).

$$m(x,y) = \sqrt{{I_x}^2 + {I_y}^2} \approx \max\big((0{,}875a + 0{,}5b), a\big) \qquad (1)$$
$$a = \max\big(I_x, I_y\big) \qquad b = \min\big(I_x, I_y\big)$$

The approximation is precise enough for the purpose of this application. Moreover, it can be completed using logical shifts, additions and comparators, resulting in minimum hardware implementation cost.

The computation of gradient directions is also performed with some simplifications. Following the ideas presented in [3] for an ASIC based implementation, the circuit for the computation of the gradient direction was modified to avoid the implementation of the costly atan2 function. Although the function can be computed using FPGAs with the CORDIC (coordinate rotation digital computer) algorithm, the proposed solution uses far less resources and does not introduces any significant performance penalty as stated in [3]. The bin is assigned based on two gradient component values as a result of a series of fixed point multiplications and comparisons, with no division necessary. Since exact, bilinear partitioning between gradient direction histogram bins is not necessary, and can be replaced by casting the equal votes on two closest bins without accuracy degradation. In fact, we go one step further than presented in [3]. If the computed orientation is within ±5 degrees of the bin's center, a double vote is cast on this bin instead. This results in a less coarse quantization. The cast votes are weighted with their respective gradient magnitude values. The results are then passed to a block that arranges them into a full cell – a 8×8 pixel structure. The corresponding values from all 64 locations are then added, with a separate adder tree dedicated to each of the nine bins. The final result is a vector of nine values, forming the descriptor of a single cell.

The descriptors from single cell processing are then passed to a functional block that organizes the cell into a grid structure used by HoG. Please note, that the building blocks of this part of the design can be configured to form a grid whose size and proportions are highly configurable and limited only by the available hardware resources as shown in Fig. 3.

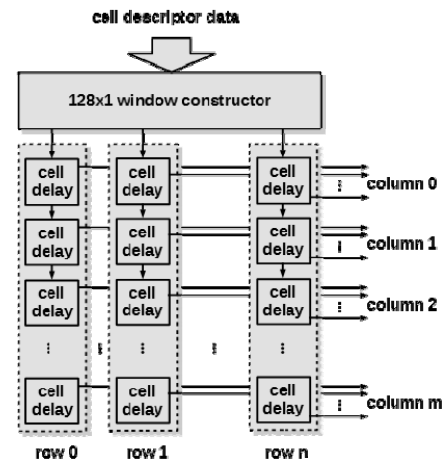


Fig. 3. Block diagram of the functional block used to arrange the block descriptors into a predefined grid structure

4. Results

The design was tested on a midrange Kintex 7 device (XC7K325TFFG900-2) from Xilinx on the KC705 evaluation platform. The maximum clock frequency reported by the

implementation tool was over 300 MHz. The power consumption is estimated to around 7 W – significantly less than a typical PC.

Resource usage for the block descriptor computation module and other elements of the system is given in the Tab. 1. To ease the analysis both, the absolute, and the relative (to the FPGA device used) resource usage is given.

Tab. 1. Resource usage of the implemented design (designations: FFs - flip-flops, LUTs - lookup tables, BRAMs - block RAM memory blocks). The values in percent are given with respect to all corresponding resources available in XC7K325T device

	LUT	% LUT	FF	% FF	BRAM	% BRAM
Block descriptor computation	6513	3.20	12612	3.09	6	1.40
Data_8_ticks_delay	216	0.11	292	0.07	0	0.0
Data_64_ticks_delay	1512	0.74	2020	0.50	0	0.00
16× Data_64_ticks_delay	24192	11.87	32320	7.93	0	0.00
Window_preparer_1_128	44	0.02	18429	4.52	254	57.08
Whole_system	30749	15.09	63361	15.54	260	58.43
XC7K325TFFG900	203800		407600		445	

The "whole system" row describes sample system, that is operating on 128×64 patch of the image and consists of a block descriptor computation module, one window prepare module and 16 modules that delay the data by 64 ticks each. Please see Fig. 3 for exact system configuration.

The main computation module (block descriptor computation) uses modest amount of FPGA logic (around 3% of FFs and LUTs), while elements responsible for the full cell arrangement are requiring large amounts of resources, especially BRAM modules. This is the most limiting factor when scaling up the application.

In Tab. 2 the performance of the system is given.

Tab. 2. Number of frames per second that the proposed hardware accelerator can process for different image sizes

Resolution	Processing time, ms	Frames per second
256 × 256	0.22	4577
16 x 256 × 256	3.49	286
64 x 256 × 256	13.98	72

First row gives the processing time for a single patch of size 256×256 pixels, in which, for every pixel, the whole HoG descriptor is calculated. The achieved 4577 fps means, that if there is only one region of interest of the said size, then the system can work with that amount of frames per second. As this is not the real-world scenario, some additional ones, with more regions to process are presented. In case of 16 regions the system achieves 286 fps and for 64 regions to process the system is capable of running at 72 frames per second. A high speed PC CPU-based implementation needs more than 3 ms to compute the response for a single detection window [6].

5. Conclusions

The paper presents a dedicated, high performance hardware architecture for HoG image region descriptor computation in image sections narrowed down to the region of interest by prior processing. It decisively outperforms the implementations running on desktop-grade CPUs while consuming less power. Moreover, thanks to the use of programmable logic, the proposed circuit can be integrated with other dedicated hardware coprocessors. It also compares favorably with other programmable hardware implementations of HoG in terms of design flexibility without compromising much of the performance and accuracy.

Future work will be focused on the integration of the described hardware coprocessor with functional blocks implementing complete classifiers and other image and video processing realizations in programmable hardware to form a complete video processing system.

6. References

[1] Dalal N., & Triggs B.: Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 1, pp. 886-893). IEEE, June 2005.

[2] Komorkiewicz M., Kluczewski M., & Gorgon M.: Floating point HOG implementation for real-time multiple object detection. In Field Programmable Logic and Applications (FPL), 2012, Aug., 22nd International Conference on (pp. 711-714). IEEE.

[3] Chen P. Y., Huang C. C., Lien C. Y., & Tsai Y. H.: An efficient hardware implementation of HOG feature extraction for human detection. IEEE Transactions on Intelligent Transportation Systems, 15(2), 656-662, 2014.

[4] Suleiman A., & Sze V.: An Energy-Efficient Hardware Implementation of HOG-Based Object Detection at 1080HD 60 fps with Multi-Scale Support. Journal of Signal Processing Systems, 84(3), 325-337, 2016.

[5] Gajski Daniel D.: Principles of digital design. Prentice Hall (1997).

[6] Kim, S., & Cho, K.: Fast Calculation of Histogram of Oriented Gradient Feature by Removing Redundancy in Overlapping Block. J. Inf. Sci. Eng., 2014, 30(6), 1719-1731.

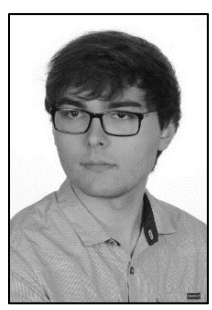
Received: 23.02.2017 *Paper reviewed* *Accepted: 04.04.2017*

Marek KRAFT, PhD, eng.

Graduated from Poznan University of Technology in 2005. He received the PhD degree from the same University in 2013. In the same year he has been appointed as an assistant professor at the university's Institute of Control and Information Engineering. His research interests are computer vision, embedded systems, robotics, parallel processing and high performance computing.

e-mail: marek.kraft@put.poznan.pl

Michał OLEJNICZAK, MSc, eng.



Graduated from the Faculty of Electrical Engineering, Poznan University of Technology, in 2016. His research interests include application of machine learning and artificial neural networks in computer vision.

Michał FULARZ, MSc, eng.

Graduated from the Poznan University of Technology in 2010, specializing in Robotics. Since then he has been employed at the Institute of Control and Information Engineering of the Poznan University of Technology. His research focuses on accelerating compute-intensive algorithms using FPGAs, image processing, embedded systems and robotics.

e-mail: michal.fularz@put.poznan.pl