FPGA Implementation of a HOG-based Pedestrian Recognition System

Sebastian Bauer, Ulrich Brunsmann, Stefan Schlotterbeck-Macht

Faculty of Engineering
Aschaffenburg University of Applied Sciences, Aschaffenburg, Germany

{sebastian.bauer, ulrich.brunsmann, stefan.schlotterbeck-macht}
@fh-aschaffenburg.de

With respect to road crash statistics, on-board pedestrian detection is a key task for future advanced driver assistance systems. In this paper, we describe the implementation of a real-time pedestrian recognition system that combines FPGA-based extraction of image features with a CPU-based object localization and classification framework. In terms of features, we have implemented the Histograms of Oriented Gradients (HOG) descriptor that is state-of-the-art in the field of human detection from a moving camera. While past HOG-related publications presented simplified FPGA-based HOG variants, often sacrificing classification performance, we implemented the original descriptor with minor modifications on dedicated hardware. Evaluation on the INRIA pedestrian database shows potential for deploying the system in practice. The descriptor computation runs on a PCIe frame grabber with embedded FPGA that can be directly integrated into an automotive computer of a test vehicle for evaluation purposes.

1. Introduction

1.1. Problem Scope

According to the preliminary report of the German Federal Statistical Office [Destatis 2009], 654 pedestrians were killed in road traffic crashes in Germany in 2008, a percentage of 15% of all road traffic related deaths (4,482). The global impact of the problem can be deduced from the first major international report on road traffic [WHO 2004]. Worldwide, an estimated 1.2 million people die in road traffic crashes annually, while the number injured could be as high as 50 million. According to model predictions, between 2000 and 2020, there will be a global increase of 67%. The majority of deaths are currently among vulnerable road users (VRU): pedestrians, pedal cyclists and motorcyclists. This is especially true in low-income and middle-income countries because of the greater variety and intensity of traffic mix and the lack of separation from other road users. In view of the economic impact, the direct costs of global road crashes have been estimated at more than US$ 500 billion per annum [WHO 2004].

1.2. Recognition of Traffic Participants

Advanced driver assistance systems (ADAS) have the potential to save numerous lives, for a survey see [Gandhi 2007]. Future pedestrian recognition systems will detect the person, predict the collision risk and warn the vehicle's driver or engage automatic braking, manoeuvring or safety devices. In the past decade, automotive night vision systems have been introduced as optional equipment on few premium vehicles. These systems are either based on near infrared (NIR, active) or far infrared (FIR, passive) sensors and increase the vehicle's driver perception distance in darkness. Traffic participants can be recognized and distinguished early when they are all but invisible by the vehicle's headlights. However, to our knowledge there is no ADAS commercially available yet that is able to recognize pedestrians in daylight situations.

Owing to the variety of possible applications in the field of surveillance, robotics and intelligent vehicles, among others, detection of humans in general has driven a surge of interest from the computer vision community over the past years, see [Gavrila 1999], [Moeslund 2006], [Poppe 2007]. As we focus on automotive applications, we use the term pedestrian in the remainder of the paper.

Vehicle-based pedestrian recognition systems are one component of the solution, but visibility from the vehicle is limited. We are developing an infrastructure-based system for ensuring crossroads safety. Such systems could complement vehicle-based hardware by monitoring the traffic and transferring the information through wireless communication channels. The fusion of complementary information of both units

can increase the virtual perception range of the vehicle and provide a more complete picture of the scene, being the foundation for proper judgements and reactions of the driver.

In this work we investigate the entropy of grayscale monocular video data for the recognition of objects. Our pedestrian detection concept is also applicable to moving platforms without algorithmic modification, for instance as a future ADAS. The system is running on a personal computer equipped with a frame grabber with an embedded field programmable gate array (FPGA) that can be used as an image pre-processor. Thus the framework can be integrated and evaluated directly in a test vehicle.

Our approach is based on a sliding-window method that evaluates image sections based on the HOG descriptor that is state-of-the-art in human detection (Fig. 1). A major contribution of this paper is the implementation of the descriptor on dedicated hardware with minor modifications compared to the scheme originally proposed by Dalal&Triggs [Dalal 2005]. While maintaining a similar classification performance level, the descriptor computation outperforms existing approaches in terms of speed. Besides, FPGA implementations are applicable for automotive applications thanks to established low cost production flows. To the best of our knowledge there has been no FPGA implementation of a HOG-based pedestrian recognition system presented in literature before.

The descriptor computation for the entire image is performed on a Xilinx Spartan 3 XC3S 4000 for the most part, leaving only a final normalization step to the CPU. Classification is performed on a graphics processing unit (GPU).

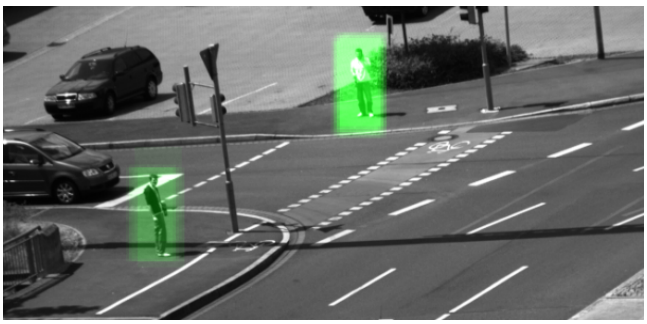


Fig. 1: Pedestrians detected with our HOG-based detection system.

The paper is organized as follows: We briefly discuss previous work in section 2. The description of the method in section 3 is followed by implementation details in section 4. Experimental results are presented in section 5, before we conclude with a summary and discuss open issues in section 6.

2. Related Work

2.1. State of the Art

Human detection in images is a challenge because of the wide variability in appearance that results from different poses and clothing, illumination conditions and complex backgrounds that are common in outdoor scenes. The excessive amount of interest in pedestrian detection led to a broad variety of different approaches in terms of system architecture, feature concepts and classification schemes.

In terms of architecture, systems can be divided into two major strategies. Part-based approaches [Ioffe 2001], [Felzenszwalb 2005], [Mikolajczyk 2004] employ individual detectors to locate single body parts that are interpreted as a human if they are arranged in a geometrically plausible configuration. Holistic approaches use a single detection window that is shifted over the image at dense positions and scales. Well known sliding-window systems rely on shape-based detection (comparison of edge images to a hierarchical exemplar dataset) [Gavrila 2007], Haar wavelet feature sets [Papageorgiou 2001], rectangle filters [Viola 2005] or the periodicity of the human walk [Fardi 2006]. Authors of both methods have offered a wide range of feature sets and classifier concepts in this domain, for a survey see [Wojek 2008b].

Recent experimental studies [Dollár 2009], [Enzweiler 2009] show that Histograms of Oriented Gradient descriptors are robust features for human detection and leading edge in terms of classification performance. Dollár et al. benchmarked seven promising pedestrian detectors on the Caltech Pedestrian Dataset [Caltech], providing an overview of state-of-the-art performance. All detectors are based on a sliding-window strategy, all use variants of HOG or Haar features. The experiments show that HOG remains competitive even on this challenging dataset. It is keeping pace with the MultiFtr and FtrMine feature sets that tend to outperform all other methods surveyed [Dollár 2009]. However, the latter two descriptors cannot compete with HOG in terms of computational load.

Enzweiler&Gavrila provide an extensive survey of pedestrian detection systems and also evaluate state-of-the-art detection systems. They use a database that is made publicly available as the Daimler pedestrian detection benchmark set [Daimler]. The experimental results show that the HOG approach outperforms all other systems they surveyed in terms

of classification performance at intermediate pedestrian image resolutions (48x96 pixels).

2.2. Histograms of Oriented Gradients

The HOG descriptor was introduced by Dalal&Triggs in 2005 as a feature set for object recognition tasks. At that time, this novel descriptor outperformed existing feature sets for human detection significantly. The basic idea is that local object appearance and shape is characterized by the distribution of local intensity gradients or edge directions, without precise knowledge of the corresponding gradient or edge positions [Dalal 2006].

The most important requirement for a pedestrian recognition system is the demand for real-time operation. However, according to Dalal&Triggs, the runtime of the object detection system lies in the range of a second for a 320x240 pixels image using a support vector machine (SVM) as classifier. For an image with twice the width and height, Dollár et al. report 13.3 seconds in their benchmark report. There have been several different approaches to speed up the HOG framework. Zhu et al. [Zhu 2006] accelerate the descriptor computation by employing an integral map (IMAP) and by replacing the SVM by a cascade-of-rejectors approach. In contrast to these software-based acceleration techniques, few authors presented HOG implementations on dedicated hardware. Besides Wojek et al. [Wojek 2008a] who take advantage of the emerging computing power of general purpose computation on graphic processing units (GPGPU), Cao&Deng [Cao 2008] presented an FPGA-based implementation of a simplified HOG framework. They realized a real-time stop sign detection system on a Xilinx Virtex-4 FPGA that is able to process 60 frames of 752x480 pixels per second by using a cascaded classifier.

3. Overview of the Method

3.1. Sliding-Window Framework

A common technique to obtain initial object location hypotheses is the sliding window scheme, where a detection window is shifted on a regular lattice over the image at various scales. For each window, a feature set (in this case HOG) is generated from the corresponding image patch and evaluated by a pre-trained classifier that categorizes unknown samples into one of the predefined classes, pedestrian or non-pedestrian in this case. Nearby detections of the same object are common with sliding-window frameworks and are typically merged using non-maxima suppression approaches in order to yield bounding boxes with confidence levels for the final detections. Sliding-window-based detection techniques showed promising performance for human recognition and have become very popular in this domain in recent literature. However, they are often considered unfeasible for real-time operation due to the immense costs in terms of resources and processing time. Significant speed-ups can be obtained by incorporating a priori information (scene geometry, target object) or by employing special classifier concepts e.g. a cascade of weak classifiers with increasing complexity. Early rejection often comes with a performance loss and methods that sacrifice classification performance to achieve speed-ups do not stand in the long term [Wojek 2008a]. As a consequence, we use a kernel support vector machine as classifier, providing the best results in Dalal&Triggs' performance study.

3.2. Descriptor Computation

The HOG descriptor is computed for each detection window with the process chain illustrated in fig. 2.

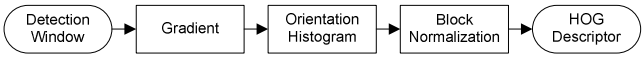
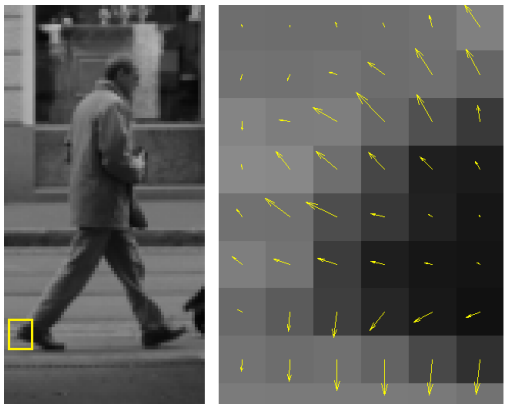


Fig. 2: HOG descriptor computation scheme.

The default detector introduced by Dalal&Triggs is based on a detection window that covers 64x128 pixels. First of all, the intensity gradient in x- and y-direction and the resulting magnitude and orientation angle is computed for the respective image patch. For an illustration of the two-dimensional gradient vectors, see fig. 3.



(a) (b)

Fig. 3a: Pedestrian example from [INRIA]. Fig. 3b: Gradients computed for an image section. The yellow arrows represent the two-dimensional gradient vector $[g_x \; g_y]^T$. Magnitude is encoded by the vector length, orientation by the vector angle.

The next step is called orientation binning and represents the fundamental nonlinearity of the HOG descriptor. The detection window is divided into 8x16 rectangular local spatial regions called cells (Fig. 4). The 8x8 cell pixels are then discretized into 9 angular bins according to their gradient orientation. Each pixel contributes a weighted vote for its corresponding angular bin, the vote is a function of the gradient magnitude at the pixel. This way the information is compressed to a 9-dimensional space per cell. The angular histogram bins are evenly spaced over 0°-180°.

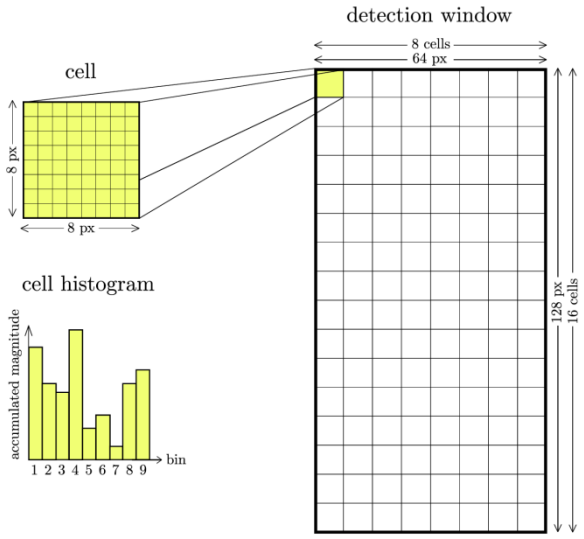


Fig. 4: Division of detection window into cells. Histogram generated for a single cell.

Gradient strengths vary over a wide range due to shadows, local variations in illumination and foreground-background contrast. Therefore local contrast normalization is essential for good performance. For this purpose, groups of 2x2 adjacent cells are considered as spatial regions called blocks. Each block is represented by a concatenation of the corresponding four cell histograms, resulting in a 36-D feature vector that is normalized to unit length, using the L2 norm.

The final HOG descriptor is represented by a concatenation of all these normalized block responses. In fact, blocks typically overlap with each other in a sliding-window fashion so that each cell response appears several times in the final feature vector, each normalized with respect to a different block. The default block stride is 8 pixels (1 cell), resulting in a fourfold coverage of each cell. To summarize: Each detection window is represented by 7x15 blocks, a block consisting of 2x2 cells, a cell is represented by a 9-bin histogram, giving a total of $(7x15) \cdot (2x2) \cdot 9 = 3780$ features.

3.3. Classification

The generated HOG descriptor is used to categorize the image patch into one of the predefined classes, pedestrian or non-pedestrian. For this classification step, Dalal&Triggs employ a support vector machine (SVM) that learns an implicit representation of the classification object from examples. In case of HOG, the classifier is pre-trained with the 3780-dimensional descriptor that is generated for all positive (pedestrian) and negative (non-pedestrian) samples, available from established datasets.

There are many different classifier concepts in the domain of supervised learning. Owing to the popularity in literature - in particular for pedestrian recognition approaches, see [Gandhi 2007], [Munder 2006] for a comparison - we also decide in favour of support vector machines.

4. Implementation

4.1. System Overview

The proposed system is based on an FPGA-CPU-GPU network. We outsourced the corpus of the HOG descriptor computation to an FPGA. The descriptor normalization step is then performed on the CPU, being the central entity of our system. The SVM-based sliding-window evaluation is currently running on a publicly available GPGPU solution [cuSVM]. An overview of the framework is shown in fig. 5. This paper focuses on the FPGA-based HOG descriptor generation.

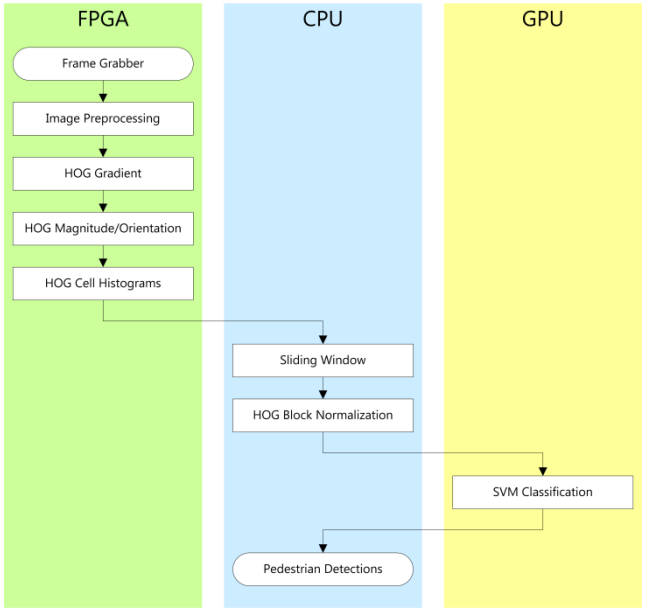


Fig. 5: Overview of the implemented FPGA-CPU-GPU framework.

4.2. Rapid Prototyping Platform

Our implementation is developed and runs on the commercially available microEnable IV-FULLx4 image acquisition and pre-processing frame grabber board from [SiliconSoftware]. It is equipped with two FPGAs, in our case a Xilinx Spartan 3 XC3S 2000 and a Spartan 3 XC3S 4000. The former provides data transfer interfaces to the camera (CameraLink) and to the host (PCIe), the latter and on-board memory is available for programming individual real-time image processing applications. The block diagram of this rapid prototyping platform is shown in fig. 6.

The board provides two external CameraLink ports that can operate with any CameraLink conform camera with Base, Medium or Full configuration. CameraLink is a serial communication protocol in the field of industrial image processing that works at a pixel frequency of 85 MHz and can transmit up to 680 MB/s by using parallel streams (taps). We use a CCD camera from [Sentech] with UXGA resolution (1600x1200 pixels) that operates at a frame rate of 30 Hz and a pixel frequency of 73.636 MHz.

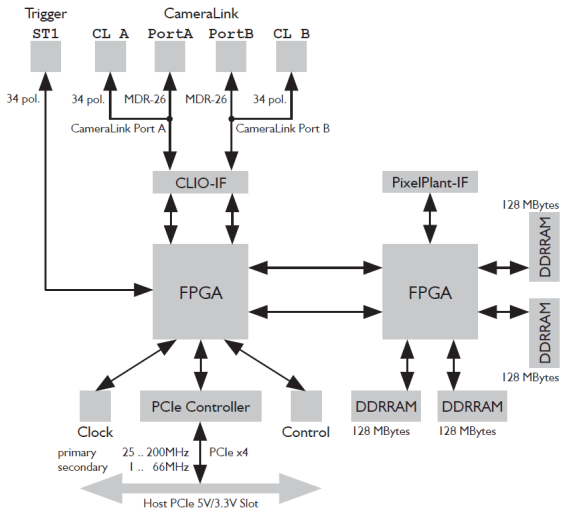


Fig. 6: Block diagram of the rapid prototyping platform microEnable IV-FULLx4 from [SiliconSoftware].

The HOG descriptor computation is implemented on the Spartan 3 XC3S 4000 that can access 4x128=512 MB on-board DDR-RAM memory. Our design is running at a frequency of 63 MHz, while parallelization enables processing of up to 32 pixels per clock cycle. For transferring the computed HOG descriptor into the main memory of the host computer, a PCIe x4 (quad lane) interface is used that achieves a bandwidth of 760 MByte/s.

With this platform, our system can be tested directly in a test vehicle by integrating the board into an automotive computer that is connected to an on-board CameraLink conform camera.

Based on a rapid prototyping idea, the Spartan 3 XC3S 4000 can be programmed with VisualApplets (VA), a graphic-oriented hardware development software from [SiliconSoftware] that is based on tools from the Xilinx ISE design suite [Xilinx]. VA enables an abstract high-level hardware implementation of customized image processing applications operating in real-time. The individual design is arranged by image processing operators and transport links that form a graphical data flow. Each operator/link can be parameterized graphically.

4.3. FPGA Design Components

Our design computes the HOG descriptor for all window positions (pixel-wise) of the entire frame. In addition to the HOG descriptor, a reference copy of the current frame is transferred through an individual direct memory access (DMA) channel. In the following, we present each individual system component. We report difficulties we faced and show our way of tackling these issues with respect to the limitations of the rapid prototyping platform.

Timing synchronization

The pixel frequency of the stream delivered by the camera is higher than the design frequency, hence buffering the image first into on-board DDR-RAM is necessary in terms of timing synchronization. Nonetheless the frequency decrease from camera to FPGA is no bottleneck, since for further processing a set of pixels can be read from the image buffer in a parallel fashion.

Scaling

With the current experimental arrangement of our infrastructure-based system, the distance of the camera to the surveillance zone is large compared to the dimensions of the surveillance zone itself. As a consequence of this arrangement, variations in pedestrian size are negligible and the descriptor is computed for one single scale level. The scale factor was set to a manually chosen value that shrinks the actual pedestrian size to the dimension of the image patches used for training [INRIA]. For applications that require multiple scale levels, the FPGA design can be modified accordingly. The maximum number of scale levels is limited by the available hardware resources.

Gradient Computation

The first step for generating the HOG descriptor is to compute the 1-D point derivatives G_x and G_y in x- and y-direction by convolving the gradient masks M_x and M_y with the raw image I:

$$G_x = M_x * I \qquad M_x = [-1\ 0\ 1]$$
$$G_y = M_y * I \qquad M_y = [-1\ 0\ 1]^T$$

On the basis of the derivatives G_x and G_y we then compute the gradient magnitude $|G(x,y)|$ and orientation angle $\phi(x,y)$ for each pixel. The gradient magnitude expresses the gradient strength at a pixel:

$$|G(x,y)| = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

We do not leave out the extraction of the square root since the performance study of Dalal&Triggs reports best results with this Euclidean metric. In our FPGA implementation, the square root operator cuts off decimal places, as we do not observe effects on classification accuracy (see section 5.1). The gradient magnitude is merely employed as a weighting factor for the orientation histogram.

The gradient orientation angle can be calculated straightforward from:

$$\tan\big(\phi(x,y)\big) = \frac{G_y(x,y)}{G_x(x,y)}$$

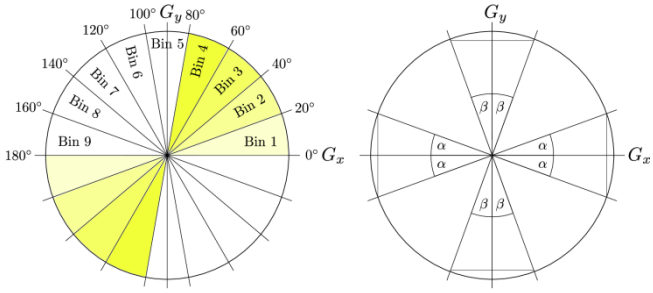
However, calculating the $\arctan()$ on an FPGA is expensive. As reported by Cao&Deng, there are hardware friendly approximation algorithms available, but they are generally iterative and slow down the system's speed. On the contrary, using lookup tables (LUTs) requires large amounts of memory which would increase system's costs. They propose to combine the gradient orientation computation with the angular binning step. Thus we are able to directly discretize the pixel's gradient angle into bins without computing the angular value explicitly.

Following this approach [Cao 2008], we introduced two improvements. We increased the number of bins from 4 to 9 in order to achieve better classification performance. In addition to that, we introduced a scheme for quantizing the pixel's gradient angle that avoids the use of signs and reduces the required bit width for relational operators.

Gradient Orientation Binning

As stated in section 3.2, the step following the gradient computation is to discretize each pixel's gradient orientation angle into 9 evenly spaced angular bins over 0°-180° (unsigned gradient), see fig. 7a. Based on the previously computed horizontal and vertical gradients G_x and G_y we first determine the angle's corresponding quadrant according to the following rule set:

$$G_x(x,y) > 0 \ \wedge\ G_y(x,y) > 0 \ \Leftrightarrow\ quadrant\ I$$
$$G_x(x,y) < 0 \ \wedge\ G_y(x,y) > 0 \ \Leftrightarrow\ quadrant\ II$$
$$G_x(x,y) < 0 \ \wedge\ G_y(x,y) < 0 \ \Leftrightarrow\ quadrant\ III$$
$$G_x(x,y) > 0 \ \wedge\ G_y(x,y) < 0 \ \Leftrightarrow\ quadrant\ IV$$



(a) (b)

Fig. 7a: Angular quantization into 9 evenly spaced orientation bins over 0°-180° (unsigned gradient). Fig. 7b: Quadrant-respective angular binning with respect to the horizontal and vertical principal axis of the unit circle.

We then compute the pixel's gradient orientation angle α with respect to its respective quadrant (0°-90°). Hence the corresponding orientation bin can be determined by quadrant and angle. The following scheme for bin 1 (0°-20°) illustrates how the $\arctan()$ computation is replaced by simple integer multiplications:

$$Cond.I:\ G_x(x,y) > 0 \ \wedge\ G_y(x,y) > 0 \ \Leftrightarrow\ quadrant\ I$$
$$\vee\quad G_x(x,y) < 0 \ \wedge\ G_y(x,y) < 0 \ \Leftrightarrow\ quadrant\ III$$

$$Cond.II: \quad 0 < \tan(\alpha) < \tan(20°)$$
$$0 < \left|\frac{G_y(x,y)}{G_x(x,y)}\right| < \tan(20°)$$
$$0 < \big|G_y(x,y)\big| < \tan(20°) \cdot |G_x(x,y)|$$
$$0 < \big|G_y(x,y)\big| < 0.364 \cdot |G_x(x,y)|$$

The multiplication in the right part of the last inequation above is a floating point operation. For FPGA implementation, the inequation is to be multiplied by a scalar >> 1 in order to replace the floating point by fixed point / integer operations. Superior is the use of bit shift operations.

The quadrant-respective angular binning is performed with respect to the horizontal and vertical principal axis of the unit circle, respectively. In case the angle α

to the horizontal axis is less than 40°, the binning is based on $\tan(\alpha)$, else on $\tan(\beta)$, see fig. 7b.

$$\tan(\alpha) = \left|\frac{G_y(x,y)}{G_x(x,y)}\right| \qquad \tan(\beta) = \left|\frac{G_x(x,y)}{G_y(x,y)}\right|$$

The advantage of this differentiation is illustrated in fig. 8. The comparison range in terms of values for condition II is reduced significantly, resulting in an equivalent reduction of bit width for the relational operators.

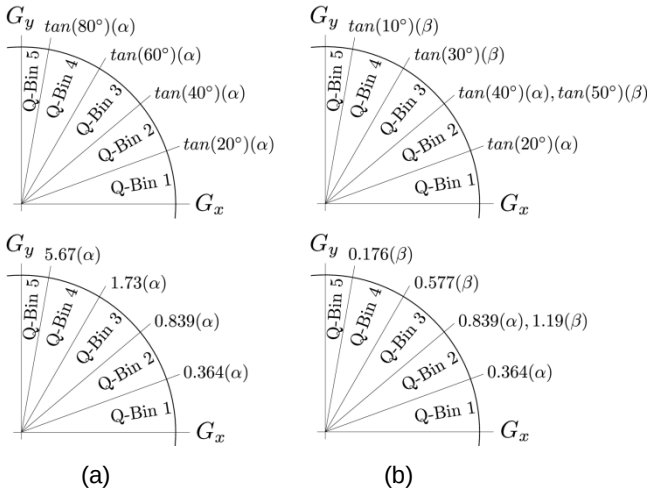


Fig. 8a: Standard angular binning of the gradient vector $[G_x(x,y)\ G_y(x,y)]^T$. Fig. 8b: Using both α and β for angular binning reduces range of values for condition II.

Histogram Generation

At this stage, we already know the angular bin (1-9) for each pixel. Then 9 binary single-channel images O_i are generated for each bin i, where the value 1 denotes that the pixel's gradient orientation lies within the corresponding angular range, 0 denoting the opposite. In a second step, we multiply each of these 9 binary bin images O_i with the gradient magnitude $|G(x,y)|$, providing 9 non-binary magnitude-weighted bin images M_i.

For each sliding window position, the histogram entry of a specific bin i in a particular cell can be easily computed by accumulating the pixel intensity values (representing the magnitude) over the cell region within M_i. In order to calculate these histogram entries for all potential sliding window positions over the entire image efficiently, recent publications work with IMAPs [Zhu 2006], [Cao 2008]. We found that this approach is not feasible with our development platform due to constraints in terms of buffers. Instead we convolute the following sum filter kernel K_S with the nine magnitude-weighted bin images M_i:

$$B_i = K_S * M_i \qquad K_S = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

4.4. What remains for the CPU/GPU

The 9 bin images B_i are to be transferred to the PC via DMA. On the CPU, the detection window is shifted over the entire image in a sliding-window fashion. The histogram entries for a specific detection window can be easily read out from the FPGA output. After that, only the final HOG descriptor block normalization step remains. At first the 4 cell histogram vectors of the current block are concatenated to a vector $\boldsymbol{v}$ with $4 \cdot 9 = 36$ components. The normalization is then performed by dividing $\boldsymbol{v}$ by the L2 norm:

$$\boldsymbol{v} \longrightarrow \frac{\boldsymbol{v}}{\sqrt{\|\boldsymbol{v}\|_2^2 + \epsilon^2}}$$

where ϵ is a small constant inhibiting divisions by zero. The values of the normalized vector are then clipped to a certain limit ([Dalal 2005]: 0.2) and the entire vector is re-normalized. This type of normalization is also known as L2-Hys.

We use a GPGPU Gaussian kernel SVM [cuSVM] for classification, operating on the normalized HOG feature vectors that are stored line by line in a matrix. The GPU can thus parallelize the classification of all windows.

5. Evaluation

5.1. Classification Performance

SVM training is performed using the INRIA dataset [INRIA] that remains the most widely used benchmark set. For the purpose of evaluation, in the following we use Dalal&Triggs' original work as a reference. Hence, a per-window evaluation is performed. Though in practice, per-window performance measures can fail to predict actual per-image performance [Dollár 2009].

In terms of positive training examples, the INRIA dataset contains 2,416 image crops (64x128 pixels) of people in roughly upright poses. 12,180 negative training examples are generated by randomly extracting image patches of the same dimension from 1,218 pedestrian-free photos. The HOG descriptors of the training examples are extracted using our FPGA-based implementation that is modified in a way that

the frame grabber module is replaced by a buffer that is able to load an image from the PC to the frame grabber's on-board memory. We performed the training process on the GPU. It took a few seconds using [cuSVM].

For evaluation, we classified 1,132 positive and 4,530 negative unseen examples (randomly sampled from 453 pedestrian-free images) from [INRIA], visualized with Detection Error Tradeoff (DET) curves on a log-log scale. DET curves plot miss rate versus false positives per window (FPPW). Miss rate is the proportion of pedestrian instances that were wrongly classified as non-pedestrian. On the contrary, FPPW is the proportion of non-pedestrian instances that were wrongly classified as pedestrian, denoted as false alarm.

We verified each individual step of our FPGA implementation by comparison to an equivalent C/C++ reference implementation running on a CPU. Fig. 9 shows that the classification performance of both our implementations is roughly consistent. In addition, DET curves from Dalal&Triggs' performance study are presented.

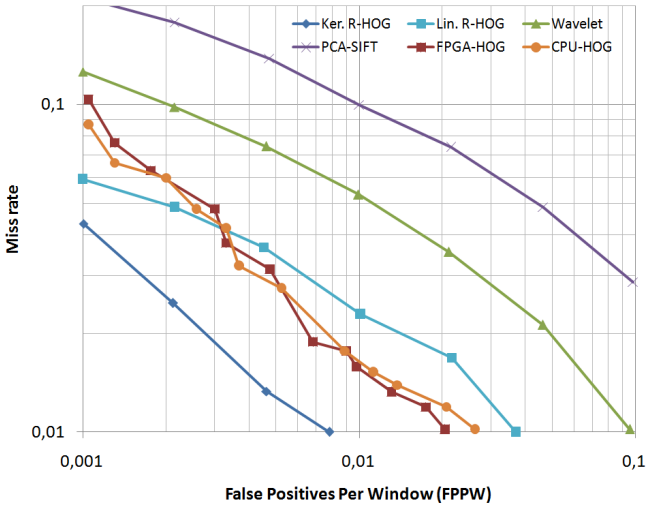


Fig. 9: Classification performance of our FPGA- and CPU-HOG implementation. The curves for Kernel SVM R-HOG, Linear SVM R-HOG, Wavelet and PCA-SIFT are extracted from [Dalal 2005].

Compared to Dalal&Triggs' original kernel R-HOG performance curve, we approach a miss rate of +6% at 10^{-3} FPPW. We deduce this difference as follows:

Essentially, in contrast to Dalal&Triggs, we have not yet applied retraining techniques such as bootstrapping that can improve the classifier performance by one order of magnitude [Munder 2006]. Furthermore, our testing set of 4,530 negative samples cannot produce positive FPPWs less than $2.2 \cdot 10^{-4}$.

We expect minor declines in classification performance due to the following implementation differences to Dalal&Triggs:

- We currently work on grayscale images, leading to a 1.5% lower detection rate at 10^{-4} FPPW compared to color images.
- Gradient magnitude is calculated with integer accuracy, i.e. all decimal places are lost when the square root is extracted.
- Our implementation does not interpolate votes between neighbouring histogram bin centres in both orientation and position to reduce aliasing effects with the histogram generation step.
- We do not down-weight pixels near the edges of the block by applying a Gaussian spatial window before accumulating orientation votes into cells. The resulting loss is about 1% at 10^{-4} FPPW.

5.2. Computation Time and Resources

With respect to computation time, we evaluate our work by measuring latency and throughput time. For latency measurements, VA operators can be used to count the number of clock cycles that elapse between the time a pixel enters a design section and the time the result is available at the section's output.

Table 1 shows latencies of the HOG descriptor computation steps, obtained by that measurement, for a UXGA camera input downscaled to a processing resolution of 800x600 pixels (SVGA) and a design frequency of 63 MHz. The histogram generation step accounts for about 2/3 of the entire latency due to the costs of convolution with the 8x8 filter kernel K_S.

Table 1: Latencies of HOG steps for a UXGA camera input downscaled to a processing resolution of 800x600 pixels and a design frequency of 63 MHz.

HOG step	Latency [µs]
Image Buffer	26.2
Scaling	26.0
Gradient	52.0
Magnitude/Orientation	0.14
Histogram	207.2
TOTAL	**311.54**

The throughput-time for one entire frame that is processed on-the-fly while grabbing the pixels is 30.8 ms for a SVGA processing resolution, the full 30 fps delivered by the camera are available via DMA.

As an indication for computation speed, one may relate the total sum of 312µs to one of the fastest implementations for HOG descriptor generation, recently published [Wojek 2008a] and running on a GPU. For the HOG steps we perform on hardware (one-time down-scaling, gradient computation, gradient magnitude, orientation binning, histogram generation) Wojek et al. report about 13 ms for an image of 320x240 pixels, computing the HOG descriptor for multiple scales.

For the SVGA image dimensions used for the latency measurements presented before, the resource usage level for the Xilinx Spartan 3 XC3S 4000 is shown in table 2.

Table 2: Xilinx Spartan 3 XC3S 4000 resource usage level for a processing resolution of 800x600 px.

Resource Type	Total number	Usage Level
4-input LUTs	42,435	76%
Internal Block RAM (18 kbit)	60	62%
Embedded Multipliers (18x18)	18	18%

6. Summary and Conclusions

As shown in section 5.2, the proposed HOG descriptor computation fits into a low-cost Xilinx Spartan 3 XC3S 4000 device. On a commercial PCIe frame grabber with embedded FPGA, it runs already fast enough that it can be directly integrated into an automotive computer of a test vehicle for evaluation purposes. The latency of the HOG descriptor generation is 312 µs and we approach classification levels in terms of pedestrian detection close to Dalal&Triggs (miss rate +6% at 10^{-3} FPPW).

Essential improvements of classification performance are expected from applying retraining techniques which have not yet been used. While Gaussian down-weighting has not yet been implemented, the histogram bin interpolation scheme does not fit into our computation approach.

Our pedestrian recognition framework based on an FPGA-CPU-GPU network currently runs in real-time in an infrastructure-based crossroads assistance system with a limited surveillance zone. The computation time for a detection window is below 150 µs, including the proportionate time for the FPGA-based HOG descriptor generation (312 µs for the entire image), CPU-based normalization (25µs) and GPU-based Kernel SVM classification (~110µs).

The system is running with 20 fps and can evaluate more than 300 detection window hypotheses per frame. This rate meets the demands of our application, as we perform a pre-selection of hypothesis windows.

There is still room to further speed up the framework. Outsourcing further parts of our recognition framework to the FPGA requires additional resources that are provided by available extension boards with additional FPGA processor and RAM. We are working on integrating the HOG descriptor normalization and SVM prediction [Irick 2008] into the FPGA, and on applying the framework for the classification of other traffic participants.

Acknowledgements

The authors acknowledge the support of the *Bayerisches Staatsministerium für Wissenschaft, Forschung und Kunst* in the context of the *Forschungsschwerpunkt Intelligente Sensorik* at Aschaffenburg University of Applied Sciences.

References

[Caltech] Caltech Pedestrian Dataset, 2009. *http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/*

[Cao 2008] T. P. Cao and G. Deng. Real-time vision-based stop sign detection system on FPGA. In *Proc. of the International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 465-471, IEEE Computer Society, 2008.

[cuSVM] *http://www.patternsonascreen.net*

[Daimler] Daimler Pedestrian Detection Benchmark, 2009.*http://www.science.uva.nl/research/isla/downloads/pedestrians/*

[Dalal 2005] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I:886-893, 2005.

[Dalal 2006] N. Dalal. Finding People in Images and Videos. PhD thesis, Institut National Polytechnique de Grenoble, 2006.

[Destatis 2009] Statistisches Bundesamt Deutschland, 2009.

[Dollár 2009] P. Dollár, C. Wojek, B. Schiele, P. Perona. Pedestrian detection: A benchmark. In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[Enzweiler 2009] M. Enzweiler and D. M. Gavrila. Monocular Pedestrian Detection: Survey and Experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2008.

[Fardi 2006] B. Fardi, I. Seifert, G. Wanielik, J. Gayko. Motion-based pedestrian recognition from a moving vehicle. In *Proc. of the IEEE International Symposium on Intelligent Vehicles*, pages 219-224, 2006.

[Felzenszwalb 2005] P.F. Felzenszwalb and D.P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision (IJCV)*, 61(1):55-79, 2005

[Gandhi 2007] T. Gandhi and M.M. Trivedi. Pedestrian protection systems: Issues, survey, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 8(3):413-430, 2007.

[Gavrila 1999] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding (CVIU)*, 73(1):82–98, 1999.

[Gavrila 2007] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International Journal of Computer Vision (IJCV)*, 73(1):41-59, 2007.

[INRIA] INRIA Person Dataset, 2005. *http://lear.inrialpes.fr/data/human*

[Irick 2008] K.M. Irick, M. DeBole, V. Narayanan, A. Gayasen. A Hardware Efficient Support Vector Machine Architecture for FPGA. In *Proc. of the IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 304-305, 2008.

[Ioffe 2001] S. Ioffe and D.A. Forsyth. Probabilistic methods for finding people. *International Journal of Computer Vision (IJCV)*, 43(1):45-68, 2001.

[Mikolajczyk 2004] K. Mikolajczyk, C. Schmid, A. Zisserman. Human detection based on a probabilistic assembly of robust part detection. In *Proc. of the IEEE European Conference on Computer Vision (ECCV)*, pages I:69-82, 2004.

[Moeslund 2006] T.B. Moeslund, A. Hilton, V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding (CVIU)*, 103(2-3):90-126, 2006.

[Munder 2006] S. Munder and D.M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(11):1863-1868, 2006.

[Papageorgiou 2000] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision (IJCV)*, 38(1):15-33, 2000.

[Poppe 2007] R. Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding (CVIU)*, 108(1-2):4-18, 2007.

[Sentech] Sensor Technologies America, Inc. *http://www.sentechamerica.com*

[SiliconSoftware] Silicon Software GmbH, 2009. *http://www.silicon-software.com*

[Viola 2005] P. Viola, M. J. Jones, D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision (IJCV)*, 63(2):153-161, 2005.

[WHO 2004] World Health Organization (WHO) and World Bank. World report on road traffic injury. Geneva, 2004.

[Wojek 2008a] C.Wojek, G. Dorko, A. Schulz, B. Schiele. Sliding-windows for rapid object class localization: A parallel technique.
In *Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, 2008.

[Wojek 2008b] C. Wojek and B. Schiele. A Performance evaluation of single and multi-feature people detection. In *Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM)*, 2008.

[Xilinx] Xilinx, Inc. *http://www.xilinx.com/tools/designtools.htm*

[Zhu 2006] Q. A. Zhu, M. C. Yeh, K. T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages II:1491-1498, 2006.