FPGA-based Pedestrian Detection Under Strong Distortions

D. Tasson, A. Montagnini, R. Marzotto, M. Farenzena
eVS embedded Vision Systems srl
c/o Computer Science Park, Strada Le Grazie, 15 Verona, Italy
www.embeddedvisionsystems.it

M. Cristani
University of Verona
Strada Le Grazie, 15 Verona, Italy
marco.cristani@univr.it

Abstract

Pedestrian detection is one of the most popular computer vision challenges in the automotive, security and domotics industries, with several new approaches and benchmarks proposed every year. All of them typically consider the pedestrians in a standing pose, but this assumption is not always applicable. It is the case of embedded camera systems used for crowd monitoring or in driving assistance systems for big vehicles maneuvering. Such systems are commonly installed as higher as possible and make use of fish-eye lenses to provide a top and wide field of view. Actually, such configurations introduce both perspective and optical distortions in the image that, even when corrected, still provide stretched silhouettes that can hardly be detected by cutting-edge pedestrian detection algorithms. In this paper we focus on this scenario, showing a) that one of the most effective models for pedestrian detection, that is the Deformable Part Model (DPM), can be efficiently implemented in FPGA to dramatically speed up the computation, and b) how it can be modified for dealing with highly distorted pictures of humans. The resulting framework, dubbed Deformable Part Model for Local Spatial Deformations (DPM-LSD), gives convincing figure of merits in terms of accuracy and throughput, on a new top-view fish-eye based pedestrian dataset (dubbed Fish-Eyed Pedestrians), also comparing with widely-used competitors (standard DPM and Dalal-Triggs).

1. Introduction

With approximately more than 2230 papers since 2000, pedestrian detection is one of the hottest challenges in computer vision, which impacts heterogeneous applicative fields spanning from robotics to driving assistance. In addition, with more than 35 datasets (for a rough total of hundred of thousands image samples) it is one of the most competitive scientific topics in terms of benchmarks[1]. However, by looking at the samples of these datasets (see Fig. 1), it is to be noted that in all cases people are viewed in some stereotypical poses, which is upright, with various orientations, captured by cameras with conventional lenses. On one side, this witnesses that pedestrian detection is still an open issue, in which many typical problems like different illumination levels, occlusions, hard human poses still have to be replicated into a dataset; on the other side, this is limiting, since different camera set-ups do exist. This is the case of cameras with fish-eye lenses installed for a top-view. These are typically used when a wide field of view has to be monitored by a single camera as for crowd monitoring [3], or in driving assistance systems to support maneuvering of big vehicles for public transport, agriculture or mining [15].

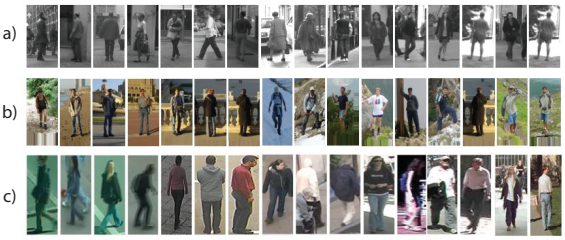

Figure 1. Some pedestrian datasets: a) Daimler http://goo.gl/HMQFav, b) Inria http://goo.gl/EY0em5, c) PETA http://goo.gl/5Pwi11.

An example of scene captured with this configuration is reported in Fig. 6, where strong distortions are easily visible, especially on the extremal region of the field of view; in addition, people under the camera are strongly

[1] See http://goo.gl/DAZVIf

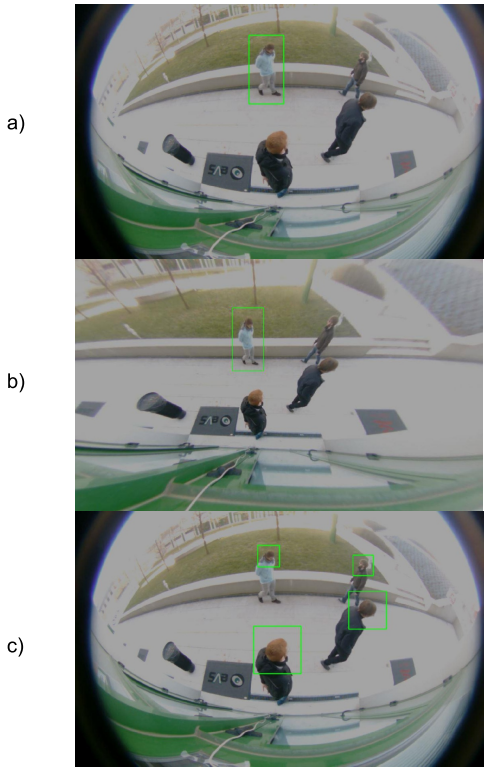

Figure 2. Fish-eye imagery: a) applying a standard Deformable Part Model [6] gives only one positive result; b) reversing the distortion does not solve the recall problem; c) our DPM-LSD gives better results.

auto-occluded, definitely different from those on the sides. Trying to apply a standard pedestrian detection algorithm on such image is very ineffective, giving poor recall performances (Fig. 6a). Correcting the lens distortion, the outcome is still unsatisfying (Fig. 6b), because the perspective distortion is still present.

In this paper, we fill this gap, presenting an embedded approach for pedestrian detection using fish-eye cameras in a top-down view. The approach considers one of the most versatile and widely adopted detector so far: the Deformable Part Model (DPM) [6]. DPM algorithm is a discriminative classification technique which models an object class as a mixture of components, where each component captures a particular visual appearance of the object (e.g the front or the side view of a car). Additionally, each component defines a number of parts encoding salient details of the object from a particular point of view (the tire or the headlight of the car). HOG features are used as low-level cue, while Latent Support Vector Machines (L-SVM) are the mixture model learning.

The first contribution of our work is the FPGA implementation of the DPM algorithm: keeping the classification performance unchanged, the FPGA version runs 120 times faster than the CPU-based, creating a general purpose, valuable engine for real-time classification. DPM algorithm is fully parallelizable and suitable to a FPGA implementation outperforming GPU-based implementations [16] by at least one order of magnitude. As a second contribution, we modify the DPM algorithm for pedestrian detection (and its FPGA-based version) in order to deal with the perspective and optical distortions: here the idea is to couple the classifier to the geometry of the camera set-up, enforcing the learning to select one particular component on the basis of the considered part of the scene; the approach is dubbed *Deformable Part Model for Local Spatial Deformations* (DPM-LSD). As third contribution, we create a dataset for pedestrian detection captured by a fish-eye camera which is composed by 4428 positive and 16100 negative samples, where the testing data is formed by 1395 fully annotated 640x480 images, named *Fish-Eyed Pedestrians*. On this dataset, DPM-LSD definitely outperforms the DPM approach and other competitors, even when they are trained on the Fish-Eyed Pedestrians dataset. An illustrative example of DPM-LSD applied on an image of our dataset is shown in Fig. 6 c).

The rest of the paper is organized as follows: in Sec. 2 the pedestrian detection approaches developed in FPGA are reviewed. This will highlight that no work in the literature deals exactly with our scenario. In Sec. 3 the DPM model is briefly summarized. In Sec. 4 the FPGA implementation of DPM is presented, while Sec. 5 describes the DPM-LSD approach. Sec. 6 will show our comparative results, also presenting the Fish-Eyed Pedestrians benchmark. Finally, Sec. 7 will conclude the paper with some remarks and future perspectives.

2. Related work

At the best of our knowledge, no other approaches in the literature deal with the detection of pedestrians in top-down distorted views. The most similar approach is the one of [3], where detection using fish-eye cameras in an indoor environment is faced with HOG features + SVM, by spanning radial stripes originating from the orthogonal projection on the floor of the top-down camera. The difference with our approach is that we are dealing with more dramatic view-changes of the pedestrians, while in their case people are essentially lying on a narrow circular stripe around the center.

As for the embedded nature of our approach, several embedded pedestrian algorithm frameworks exist in the literature. Most approaches are entirely FPGA-based (all the processing is done on FPGA), with emphasis on the design of the features – HOG-based in most of the cases [9, 11, 14] – with simultaneous SVM classification [13], on high resolution images and multiple scales [7]. Other descriptors are covariances [12] or co-occurrences of features [8]. Few

approaches perform only part of the processing on FPGA [1, 2]: even in this case, all of them focus on pedestrians with no strong distortions. GPU implementations of HOG-DPM based object detectors are proposed in [16], while no FPGA implementation of the DPM approach is available in the literature.

3. The DPM algorithm

In the following, we summarize the DPM classifier, fully presented in [6], highlighting those aspects which are more related with our approach, and adopting the same formal notation for coherence.

The DPM classifier is built over a m-component mixture model $M = (M_1, \ldots, M_m)$, where M_c is the model for the c-th component, and m is a priori defined. Each component is designed to focus on a particular visual facet of the modeled class (for example, an object in a particular pose). Dropping the index c for clarity, the component model is $M = (F_0, P_1, P_2, \ldots, P_n, b)$, where F_0 encodes the *root filter* defining the reference position of the object in the image, P_i is the model for the i-th part and b is a bias term acting as a normalizer among components. The parts encode portions of the object of interest and the number of parts n is defined a priori. The model for each part is $P_i = (F_i, v_i, d_i)$, where F_i is a filter specialized to recognize a particular visual part (arm, torso, etc. in the case of a pedestrian), v_i indicates the position of the part in relation with the root position and d_i identifies a cost for a particular deformation. In the learning stage, bounding boxes around the objects of interest in the training images initialize the root position F_0; training alternates between two steps, in which i) the components are updated and ii) the images are assigned to components.

As for the features, HOG-like cues are used, forming a pyramid which essentially connects the level where the root filter is applied to that where the parts are examined, i.e. at double resolution wrt the root level.

Once trained, the SVM model scores each test example x by a function of the following form,

$$f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \phi(x, z)$$

where β is a vector of model parameters, containing the root filter, the part filters, and deformation cost weights; $\Phi(x, z)$ is a feature vector containing a concatenation of sub-windows and part deformation features in relation with the feature pyramid, specified by z. Given an example x, all possible z configuration values are considered by adopting a distance transform approach [10], and the maximum is taken as detection result for the component. The final detection is the maximum over the components. Afterward, a post processing step is implemented to deal with multiple overlapping detection, which essentially applies a non-maxima suppression step.

During the learning, the β parameter vector is estimated using a latent SVM strategy.

4. FPGA implementation of DPM

The target device for DPM algorithm implementation is Xilinx®Zynq™-7000 All Programmable System-on-Chip (SoC) that combines a dual-core ARM®Cortex™-A9 processing system (PS) with Xilinx 28-nm programmable logic (PL) on a single device. A key advantage of this device is the partitioning flexibility offered through tight coupling between PS and PL. The pixel-level, high data rate processing can be accelerated in hardware and decision making tasks implemented in the serial processor. This approach fully offloads the serial processor (ARM Cortex), thereby avoiding processing bottlenecks and achieving throughput increases of up to several orders of magnitude. The adopted development platform is the ZC706 board equipped with a XC7Z045 device. To grab the HDMI video source we use a DVI I/O daughter card. The FPGA fabric is responsible for acquiring the video from the HDMI input port (Video Input), generating the pyramid of images (Video Scaler), performing the object detection algorithm (DPM) and streaming out the video to the HDMI output augmented with graphics/information overlay (Video Controller). The processing system is primarily responsible for configuring and controlling the FPGA-based processing blocks, and performing post-processing operations such as the non-maxima suppression.

Here we focus on the DPM core only, that is the main functional block implementing the object detector. This implementation accurately fits the original algorithm formulation of [6]. In Fig. 3 the DPM core block diagram is shown. The DPM core itself works at single image scale. For each acquired frame, the Video Scaler provides the DPM with a sequence of images obtained by resizing the input. The input stream is provided by the Scaler at double resolution so a preliminary sub-sampling stage (binning) is necessary to generate two parallel streams, the first at input resolution and the second one at half resolution. The two streams will be used to compute the normalized HOG feature for respectively the parts filters $F_0, F_1, \ldots, F_n$ and the root. DPM processes the image in raster scan fashion. The HOG feature extraction requires to compute the image gradient, calculate a 18-bins histogram of the gradient orientation of 8x8 pixels cells and normalize it. Each pixel contributes to the feature vectors in the four cells around it using bilinear interpolation. The feature dimension is 31 and each element is represented with 18 bits. Linear SVM and distance transform are processed in parallel for the root and each part of the components. Distance transform is locally computed 9x9 window of cells. This approximation was necessary to

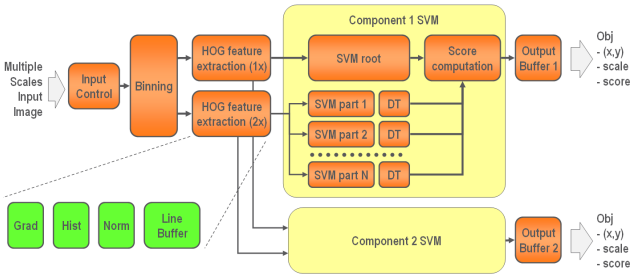

Figure 3. DPM core block diagram.

Table 1. DPM core FPGA resources occupation.

Zynq-7045	Used	Available	Percentage
Flip-Flops	78,945	437,200	18%
Look-Up Tables	58,556	218,600	27%
Block RAM	302	545	55%
DSP slices	702	900	78%

save internal memory but does not affect the accuracy from our experiments. Each SVM is implemented using an array of MACs (multipliers with accumulator) calculating in parallel the inner product between the normalized feature vector and the corresponding vector of weights. The MACs are implemented using the DSP48 slices. The number of used MACs is related to the maximum template size (12x16 cells = 96x128 pixels). The partial scores of the root and the parts are then combined to calculate the final score of each component $f_\beta(x)$. The detection window scans the entire image and, for each position, the score is computed. If the score is greater than a threshold, the current position, pyramid level and score information are inserted into a shared output buffer containing the list of detected objects. The results of the detection are than read out by the ARM processor that applies the post-processing stage. All DPM computation is done in fixed point precision except from the feature normalization (L2-norm) that is done using single precision floating point arithmetic operators. The numerical precision is accurately trimmed in order to keep the score error smaller than 0.05 with respect to the floating point software model.

The core can be configured at compilation time to modify the maximum image resolution, template size, number of parts and components. This allows to optimize the area consumption on the basis of the object model. The programmable logic resource consumption of the DPM core is estimated with ISE Design Suite 14.7 and reported in Tab. 1. This estimation is related to the DPM core with a general configuration (2 components, 6 parts and a maximum template size of 12x16 cells). The core is fully programmable via software: it is possible to load different object models and pyramids at run-time.

The generation of the image pyramid performed by the scaler requires a huge external memory bandwidth and is the actual bottleneck of the pipeline. The input pixel rate is 120 Mhz, while the clock frequency of the core is 240 Mhz because it works at double data rate. The frame rate depends on the number of image pyramid levels (= number of processed images for each acquired frame). With 28 levels and 640x512 input image resolution (corresponding to 1280x1024 into the core) the frame rate is about 11.75 fps, about 120 times faster than a PC-based implementation (Intel Xeon 2.66 GHz - 8 GB RAM). The latency is about 56 image lines. The global throughput of the core is greater than 170 Giga MAC/sec.

5. The DPM-LSD model

The Deformable Part Model for Local Spatial Deformations classifier modifies the DPM in those cases in which the visual variance of the objects to be detected is strictly related with the geometry of the scene and with the physical configuration of the sensors, and when these relations are known beforehand. This is the case of cameras in top-down view, far from the floor. In this case, going far from the projection of the center of the lens on the floor – where the visual appearance of the pedestrian is dominated by head, shoulder and feet – one can notice that the body silhouette starts to be stretched and visible in its entirety, the direction of stretching depending on the radial angle wrt the orthogonal projection of the camera (see Fig. 6). This effect is even more dramatic in the case of fish-eye lenses. In such a scenario, one could try to apply the standard DPM, letting the mixture learning mechanism distribute correctly the learning samples to the components depending on their position on the floor. Unfortunately, this does not happen, and components turn out to model not well defined compounds of visual appearances, without any relation with the physical scenario. For the sake of the classification, this compromises the overall quality of the detection.

Our idea is simple yet effective, and consists in enforcing the training samples to belong to a particular component M_i, depending on their distance r from the projection on the floor of the lens center, independently on the radial angle they exhibit. Technically, this corresponds to add a component label $I_t = M_i$ to the t-th image, initializing the assignment to the m components accordingly; therefore, these images are segregated to stay in one particular component during the whole training procedure. For each component, the parts $\{P_i\}$ are chosen automatically, so that the parameters fit the variance of the images of that component. The number of parts per component is fixed beforehand.

6. Experiments

6.1. The Fish-Eyed Pedestrians dataset

The Fish-Eyed Pedestrians (FEP) dataset is composed by pictures extracted from 640x480 video sequences cap-

Figure 4. Fish-Eyed Pedestrians: some sample positive images.

tured around the University of Verona campus, in indoor and outdoor settings. The acquisition setup was formed by a fish-eye camera, with 130° field of view lens. The camera has been set 3.5 meters over the ground. We divided the sequences into training and testing. From the training material we extracted 2220 positive and 16100 negative samples while the testing data is formed by 1395 frames containing 2208 positive samples. Some training images are shown in Fig. 4: they include people walking with friends, with backpacks, and under different illumination conditions. Besides the training samples, we also provide the distance in meters wrt the center of the scene, in order to allow the DPM-LSD learning.

6.2. Comparative experiments

In the detection experiments, evaluated on the FEP testing partition, as evaluation metrics we adopt the Jaccard similarity coefficient (JSC), which is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

where in our case A and B are the bounding boxes under exam. A bounding box is considered correct if JSC is greater than 0.5, otherwise we have a false positive detection.
As for the comparative approaches, we consider:

HOG+SVM method. We use the C++ implementation of the Dalal & Triggs detector [4]. Size of feature cells is 8×8, and the box size is 16×16. Gradient orientations are discretized into 8 values. Template dimension is 64×64. We perform SVM training by using SVM-Light, and we use the FEP training dataset.

DPM trained with PASCAL dataset. This is the method (software based) employed in [6], which uses as training set the PASCAL dataset (the positive coming from the "person class", the negative coming from an uniform sample of the other classes) [5].

DPM trained with our samples. As above, but using the FEP training dataset.

DPM-LSD. DPM-LSD model (FPGA-based), trained with the FEP training dataset, using 2 components, with 6 and 3 parts. The first component covers the central circular area with radius $r = 1.80m$, the second component models the other samples.

As for the computational speed, the results discussed in Sec. 4 still hold, with the DPM-LSD model working 120 times faster than the software DPM (the DPM-LSD extension did not bring any delay wrt the simple DPM FPGA-based model, since it basically affects only the training procedure).

In Tab. 2 we report the quantitative results of the detection performances; in Fig. 5 we plot the related Precision Recall curves (best viewed in colors).

Table 2. Average Precision values

Configuration	Average Precision
HOG+SVM	0.45
DPM-LSD 3 parts	0.62
DPM-LSD 6 parts	0.55
Standard DPM with our dataset	0.37
Standard DPM with PASCAL dataset	0.027

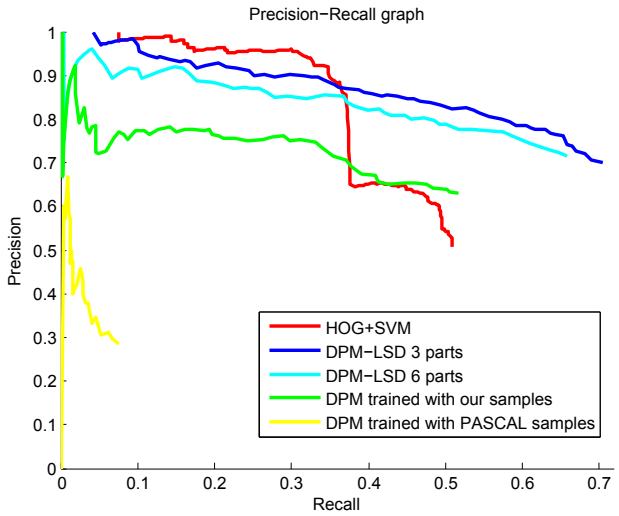

Figure 5. Precision Recall graph on the pedestrian detection task.

Intuitively, the use of the FEP dataset helps the DPM approach to increase its performance wrt a standard training dataset. Still, it is definitely inferior to DPM-LSD. HOG+SVM seems to set the best score at very low recall

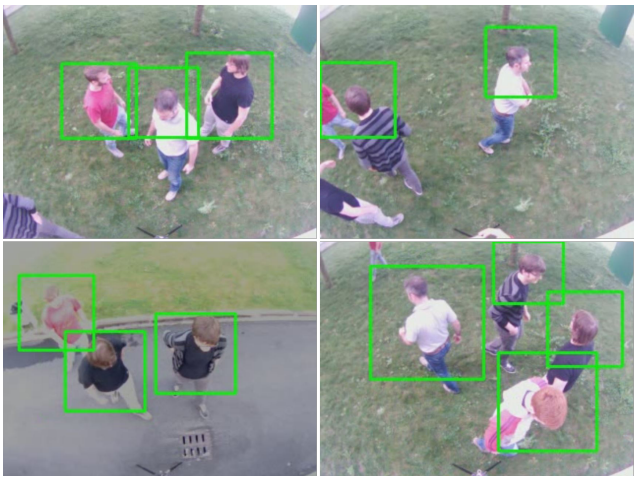

Figure 6. Examples of detection results.

levels, with a dramatic decrease at 0.37 of recall. DPM-LSD, on the contrary, is more constant, exhibiting the best curve with 3 parts. Looking at the trained models (not showed here for space reasons), the 3 parts focus on the head (one part) and the shoulders (2 parts), while in the case of 6 parts there is no easy interpretation.

7. Conclusions

In this paper we faced the novel problem of detecting pedestrians with strong distortions, considering a top-down view setting. Despite the scarce coverage in the literature, we think that this scenario is worth to be investigated, since many are the situations in which such configuration does occur. Here we offered an initial embedded solution, getting inspiration from the DPM approach, plus a novel dataset for future comparisons. Future work includes the investigation of an automatic procedure to discover the best number of components and parts.

References

[1] S. Bauer, U. Brunsmann, and S. Schlotterbeck-Macht. Fpga implementation of a hog-based pedestrian recognition system. 2014. 3

[2] J. Brookshire, J. SteffJorgenensen, and J. Xiao. Fpga-based pedestrian detection, 2010. 3

[3] A.-T. Chiang and Y. Wang. Human detection in fish-eye images using hog-based detectors over rotated windows. In *Multimedia and Expo Workshops (ICMEW), 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014. 1, 2

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1(1):886 – 893, June 2005. 5

[5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 5

[6] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 2, 3, 5

[7] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll. Fpga-based real-time pedestrian detection on high-resolution images. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 629–635. IEEE, 2013. 2

[8] M. Hiromoto and R. Miyamoto. Hardware architecture for high-accuracy real-time pedestrian detection with cohog features. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 894–899. IEEE, 2009. 2

[9] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura. Hardware architecture for hog feature extraction. In *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP'09. Fifth International Conference on*, pages 1330–1333. IEEE, 2009. 2

[10] R. Kimmel, N. Kiryati, and A. M. Bruckstein. Sub-pixel distance maps and weighted distance transforms. *J. Math. Imaging Vis.*, 6(2-3):223–233, June 1996. 3

[11] X. Ma, W. Najjar, and A. R. Chowdhury. High-throughput fixed-point object detection on fpgas. In *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, pages 107–107, May 2014. 2

[12] S. Martelli, D. Tosato, M. Cristani, and V. Murino. Fpga-based pedestrian detection using array of covariance features. In *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*, pages 1–6. IEEE, 2011. 2

[13] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto. Architectural study of hog feature extraction processor for real-time object detection. In *Signal Processing Systems (SiPS), 2012 IEEE Workshop on*, pages 197–202. IEEE, 2012. 2

[14] K. Negi, K. Dohi, Y. Shibata, and K. Oguri. Deep pipelined one-chip fpga implementation of a real-time image-based human detection algorithm. In *Field-Programmable Technology (FPT), 2011 International Conference on*, pages 1–8. IEEE, 2011. 2

[15] H. Tadjine, M. Hess, and S. Karsten. Object detection and classification using a rear in-vehicle fisheye camera. In *Proceedings of the FISITA 2012 World Automotive Congress*, volume 197 of *Lecture Notes in Electrical Engineering*, pages 519–528. Springer Berlin Heidelberg, 2013. 1

[16] Hirabayashi, M.; Kato, S.; Edahiro, M.; Takeda, K.; Kawano, T.; Mita, S. GPU implementations of object detection using HOG features and deformable models. In *Cyber-Physical Systems, Networks, and Applications (CPSNA), 2013 IEEE 1st International Conference on*, pages 106–111. IEEE, 2013 2, 3