Towards a Real-Time Pedestrian Detection based on a deformable template model

Marco Pedersoli, Jordi Gonzàlez, Xu Hu and Xavier Roca

Abstract—Most advanced driving assistance systems already include pedestrian detection systems. Unfortunately, there is still a trade-off between precision and real-time: for a reliable detection an excellent precision-recall, such a trade-off is needed to detect as many pedestrians as possible while, at the same time, avoiding too many false alarms; also a very fast computation is needed for fast reactions to dangerous situations. Recently, novel approaches based on deformable templates have been proposed, since these show a reasonable detection performance, although computationally too expensive for real-time performance.
In this work we present a system for pedestrian detection based on a hierarchical multi resolution part-based model. The proposed system is able to achieve state-of-the-art detection accuracy, due to the local deformations of the parts, while exhibiting a speed-up of more than one order of magnitude thanks to a fast coarse-to-fine inference technique. Moreover, our system explicitly infers the level of resolution available so that the detection of small examples is feasible with a very reduced computational cost.
We conclude this contribution by presenting how a GPU optimized implementation of our proposed system is suitable for real-time pedestrian detection in terms of both accuracy and speed.

Index Terms—Driving Assistance, Object Detection, Pattern Recognition.

1 Introduction

Driving assistance is a growing area of research that involves many different disciplines, from mechanics to computer science. The fields of application span not only very specific and rule based systems, like the Antilock Brake System and air-bags present nowadays in almost every commercial vehicle, but also very challenging and complex tasks, like following the correct path while avoiding obstacles and accidents in uncontrolled scenarios.

In this paper we deal with a very specific, but fundamental task, which is pedestrian detection using a single camera mounted on the vehicle. Being able to detect pedestrian as well as other objects using only a normal camera sensor would be a great technological advance. In fact cameras, compared with other sensors like laser scanner, are very economic and this would allow this technology to be deployed also in low class vehicles.

From a profane point of view the task of detecting and localizing pedestrians from single images looks simple. We can find in every last-generation photographic camera a face detection system that works pretty real-time. So, theoretically, the task is just about learning pedestrians instead of faces. Unfortunately the task is not so easy because: (i) pedestrians have a much broader appearance variability than faces or other objects [1] thus traditional methods like [2], [3] do not really work for pedestrians (ii) even if we are able to achieve the same performance than for faces, the real application is much more error critical in the sense that a false detection in a pedestrian avoidance system would produce an abnormal vehicle behavior (e.g. automatically braking when not necessary) as well as missing a detection can lead to a dangerous situation (e.g. risk of run over).

• *Marco Pedersoli, Jordi Gonzàlez, Xu Hu and Xavier Roca are with Computer Vision Center and the Universitat Autònoma de Barcelona, Catalonia (Spain). Contact e-mail: marcopede@cvc.uab.es*

The computer vision community in the last years has developed better methods that achieve higher accuracy and can deal with more challenging and complex objects categories [4], [5]. However this level of accuracy has been reached at the cost of relaxing the real-time performance requirement, due to the higher computational cost of complex features [4] and complex object models [5].

Restricting detection to the specific task of pedestrian localization for driving assistance, some speed-ups can be achieved. For instance, in normal conditions, the upper part of the image always contains sky and the search for pedestrians is usually avoided, thus producing a save in time as well as in number of false positives. Also, more sophisticated techniques to reduce the number of location to scan using specific knowledge of the problem can be used [6]. Furthermore, as we deal with video sequences it is possible to add temporal coherence among frames, that can contribute to further reduce false positives and localize the search to specific image regions.

Still, considering that the time for computing an image in a high-level PC is in the order of one minute for [4] and around 10 seconds for [5], they are too far for reaching real-time performance. Furthermore, these techniques work properly when the object to detect has a relatively high resolution. This condition is not satisfied in pedestrian detection for driving assistance, where it is very important to detect far, low resolution, pedestrians.

In this paper we present a framework for pedestrian detection that merges recent advances on deformable object detection with strategies specialized for the task of pedestrian detection from a moving vehicle. The contributions of this work are the following: i) we use a multiresolution representation and the coarse-to-fine strategy proposed in [7] to speed up the search for pedestrians in an image ii) we introduce an additional reasoning about small objects that normal detectors can not detect: scores of small objects where high resolution features are not available, are made comparable to full-resolution detections, adding an additional bias iii) we speed up the feature computation, which is the bottleneck of our method, by computing them on a GPU. Altogether, we show that our final system is suitable in terms of both accuracy and speed, for real-time applications.

The structure of the paper is the following. In section 2 we consider the related work, whereas in 3 we present an overview of our detector system. More in detail, we formulate the coarse-to-fine search (sect. 3.1), we extend it to deformable templates (sect. 3.2), we introduce the resolution features to detect small pedestrian (sect. 3.3), and finally we formulate the latent SVM problem (sect. 3.4). In section 4.1 we adapt the system for real-time computation by computing the image features on a GPU and by considering a specific region of interest. Finally, in section 5 we evaluate the performance of the system on a pedestrian specific database, and in section 6 we give the final conclusions.

2 Related work

In driving assistance, reliable pedestrian detection is an essential but very difficult to achieve. Considering the additional constraint that the system should be real-time, building such a system is very challenging. In past years, the common way to obtain a good compromise of speed and detection accuracy, was to make use of multiple sensors, in particular stereo cameras [8].

An early example of a real-time system for pedestrian detection was presented in [9], where the authors coupled stereo measurements computed from a couple of cameras mounted on the vehicle together with a detection system based on neural network classification. From disparity images the algorithm selects only a reduced set of candidates regions that will be subsequently classified using disparity and appearance cues.

In [10] a pipeline of different cues based on stereo vision is used to obtain a fast an accurate pedestrian detection. A survey of different algorithms for real-time dense stereo for pedestrian detection is presented in [11]. A system that integrates detection from stereo images and tracking using particle filters is presented in [12]. Finally, in [13] using stereo images, the ground plane of the road is estimated to search for the pedestrian in a limited region of the image.

In more recent years, object detection has shown great improvements and open the way to the possibility to detect pedestrians using single images (monocular cameras). Current object detection has two main families of methods to tackle the problem: bag of words (BOW) and template matching (TM) models. The first one comes from document classification techniques and claims that a powerful way to distinguish different document categories is to learn statistics about the distribution of relevant words in the text. Now for the task of image classification words have been substituted by quantized visual features (i.e SIFT [14]), but the main idea remains the same. Examples of this method for image classification can be found in [15], [16], [17], [18] while examples of detectors in [4], [19]. The second approach for object detection is based on an even simpler representation: an object is represented by a learned template, i.e the average of gradients of a collection of objects of the same category. Recent examples of detectors based on this technique can be found in [20], [21], [22], [5].

In both representations, in order to localize an object in an image, it is necessary to scan the learned object model at all possible scales, positions, (and depending on the problem also rotations) and evaluate how similar is the model to the local image location. Considering that in a standard image we can find millions of possible locations, it is easy to see that for object detection a fundamental feature is to use a fast way to evaluate the similarity between object model and image region.

Although a lot of work have been done for speeding-up BOW techniques [23], [24], [25], these are intrinsically slower than TM techniques. In particular, while in TM techniques the image features are directly evaluated for computing a similarity measure, in BOW based techniques a further step of quantization is applied after the feature extraction, with a computational time of the order of $O(WF)$ where F is the number of features extracted and W is the number of words used. Generally for good performance both factors F and W are on the order of thousands, therefore this implies a very high computational cost. In contrast TM techniques do not compute any feature quantization and in the case of linear classifiers, their computational cost is $O(F)$ proportional to the feature size.

Although BOW models are possibly a more powerful representation, so far only TM are suitable for real-time performance. In this sense, in [26], [27] two similar methods for real-time object detection are presented. They use integral-image for fast computation of gradient-based features and boosting for learning. Although quite fast, these methods do not consider part deformation, which is important for getting state-of-the-art accuracy. Recently, Benenson et al. [28] have improved [27] introducing scale specific detectors and a GPU implementation to get real-time performance.

In [29] a cascade of parts to speed-up deformable object detection is proposed. The method achieves a similar speed-up to the coarse-to-fine procedure and it can be used in conjunction with the coarse-to-fine procedure [7] to further speed-up the detection phase.

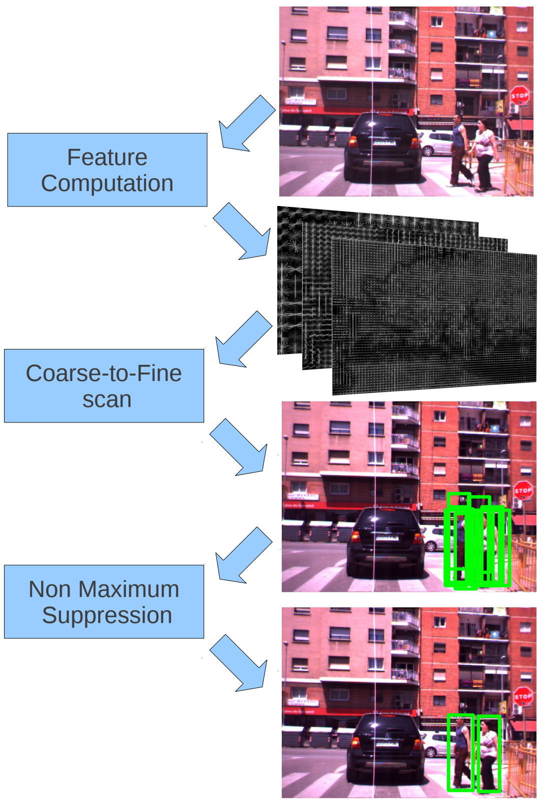

Fig. 1. Overview of our detection system: From the image a pyramid of HOG features is computed and given as input for the CtF procedure, which finds the best detections. These detection are subsequently filtered using a non maximum suppression procedure.

Finally, in [30], the authors propose to use GPU computation and ground-plane constraints to also obtain a real-time system for pedestrian detection. Also in this case, the lack of deformations in the model reduces the detector accuracy and, as it will be shown in section 5, make it too weak for real-world applications.

For a complete survey on pedestrian detection and datasets see [31], [32] and [33].

3 Our System

The architecture of our system for pedestrian localization is illustrated in Fig. 1. Given an image, we precompute the HOG features of the image at different resolutions, obtaining a pyramid of HOGs. Then, given an object template or model, the pyramid is scanned at all resolutions in a Coarse-to-Fine way, finding the locations that are the most similar to the template. These locations are further processed by applying a non-maximum suppression (NMS) to the overlapping ones. The remaining locations, represent the detected pedestrians. In the following subsections we will explain the most relevant parts of our system. For the HOG computation we use the implementation of [5] which is an improvement over the standard HOG features [20]. For NMS we rank the detection scores and we select the 1000 best detections for a greedy clustering based on the PASCAL overlapping criteria [34].

3.1 Coarse-to-Fine search

The standard procedure to find an object in the image consists of evaluating the similarity between the object model and the image features at every location and scale in the image. Considering that we use a model learned with linear SVM, the similarity measure is the scalar product of the object model M and the corresponding pyramid feature H at location $\mathbf{x} = (x, y, s)$, where x, y are the coordinate of the window center and s is its scale. Therefore, the standard search is the correlation between M and each level of the HOG pyramid:

$$D(\mathbf{x}) = \langle M, H(\mathbf{x}) \rangle \, . \tag{1}$$

In general, fine feature resolutions are required to learn more discriminative detectors, which also produces a more complex object model. This implies that the vectors of the scalar product in Eq. (1) can be of the order of thousand dimensions, therefore, the complete scan over positions and scales is very expensive and often it is the computational bottle-neck of the entire system. Facing this problem, authors in [35] proposed a Corase-to-Fine (CtF) search to save computation but still obtain results very similar to the complete search.

The key idea is to decompose the search over multiple resolutions: from coarse and then fine. The coarse resolution has less locations where to scan and the scalar product is faster to compute because the vectors have less features. However, few coarse features are not enough for good discrimination of the model. For this reason adding finer resolutions improves performance but increases the computational cost. In practice we can think the CtF procedure like a progressive refinement search: the coarse object representation is used to roughly and quickly find the object locations and then successive local refinement are applied with the next model resolutions. Fig. 2 illustrates the CtF procedure.

The score of the multiple resolution detector is computed as sum over resolutions r of the object model M_r with the corresponding features H:

$$D(\mathbf{x}) = \sum_{r=1:R} \langle M_r, H(\mathbf{x}_r) \rangle \, . \tag{2}$$

Considering a model resolutions with scale ratio equal to 2 (i.e. each model M_{r+1} doubles the previous model resolution M_r), we set :

$$\mathbf{x}_r \rightarrow \mathbf{x}_{r+1} = 2\mathbf{x}_r \tag{3}$$

to impose that the locations x_r for all resolutions r represent the same image position. An example of a 3-resolutions object model for pedestrians is shown in Fig. 3(a).

In the coarse-to-fine procedure, the search starts computing the score of the coarse model everywhere in feature space $\mathbf{x}_1$. After that, the score locations are clustered

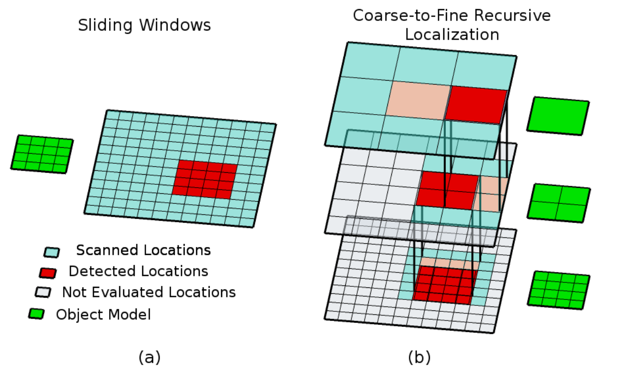

Fig. 2. Comparison of a standard sliding windows scan versus the CtF procedure. (a) Sliding Windows has to evaluate all locations at high resolution. (b) CtF localization evaluates the image everywhere only at coarse resolution, then the best hypotheses are propagated and evaluated only locally producing an high computational saving.

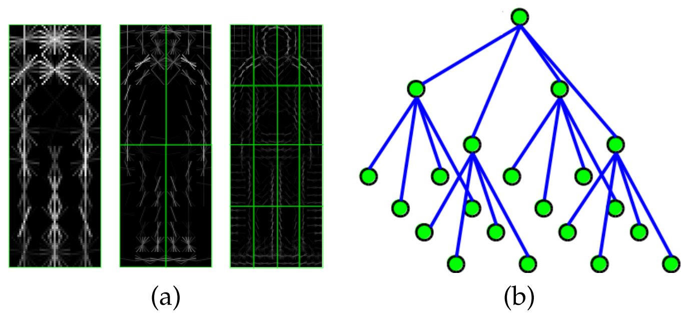

Fig. 3. (a) Example of a multiple resolution deformable part model: each part is a collection of HOG filter at different resolution. (b) The HOG filters form a father-child hierarchy where connections control the relative displacement of parts.

into local neighborhoods, and like in NMS only the highest score for each neighborhood is selected. The selected hypotheses are propagated to the following resolution model M_2 using Eq. (3). Now again, a local neighborhood is built around every hypothesis. Notice, that after the first resolution level, the local neighborhoods do not cover anymore all the possible locations of the image, but only a small fraction around the hypotheses. This produces a high computational saving because the scalar product has a computational cost that increases 4 times when doubling resolution, but with the CtF procedure only a small fraction of locations is actually computed: those that are close to the hypotheses. The procedure is then recursively repeated for all model resolutions.

3.2 Coarse-to-Fine search with deformations

In this section we extend the Coarse-to-fine search to deformable parts models. Adding moving parts allows the detector to better adapt to local object deformations that are produced by view point changes or articulated movements, like limbs movements in the case of pedestrians. Unfortunately adding deformation to the object model supposes a huge increment of computation because for each location the best object parts configuration should be found. Previous methods reduce the computational cost of finding the best object parts configuration using distance transform, assuming squared deformation cost [5]. This procedure reduces the cost of matching parts, but still, all locations should be evaluated by computing the costly dot product between features and object model.

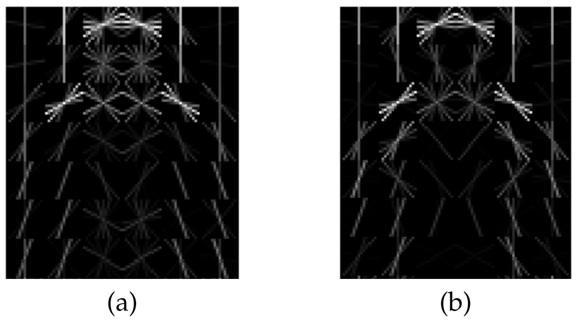

Fig. 4. Detail of a pedestrian head and torso model. (a) Model learned *without local deformation* (b) Model learned *with local deformation*. The second model has clearer edges due to the local deformations.

To reduce this cost, we use the CtF procedure extended to deformable models presented in [7]. Now, in the object model, each resolution level is further divided into parts as shown in Fig. 3(a) (green boxes). Specifically, the coarse representation of the model has only one part. The middle resolution is divided into P local parts, and moving to the finer resolution, each of these parts is again decomposed into P sub-parts in a recursive way, creating a tree-like structure as shown in Fig. 3(b). The object score at a certain location $\mathbf{x}$ is now computed as:

$$D(\mathbf{x}) = \sum_{r=1:R} \sum_{p=1:P^{r-1}} \left\langle M_{r,p}, \left[H(\mathbf{x}_{r,p}), d_x^2, d_y^2\right] \right\rangle \quad (4)$$

where the feature vector is now extended with the deformation features d_x, d_y that represent the displacement of a part p with respect to its father. In the CtF procedure with the new object model, the initial hypothesis produced by the model $M_{1,1}$ is propagated to the next resolution level generating P new hypotheses for the sub-parts:

$$\mathbf{x}_{r,p} \rightarrow \mathbf{x}_{r+1,1}, \mathbf{x}_{r+1,2}, ..., \mathbf{x}_{r+1,P-1} \quad (5)$$

The procedure is recursively repeated until covering all parts of the model. Each new hypothesis $\mathbf{x}_{r+1,i}$ is found at double resolution and with a certain offset $\mathbf{o}_i$ due to the relative sub-part location:

$$\mathbf{x}_{r+1,i} = \mathbf{o}_i + 2\mathbf{x}_{r,p}. \quad (6)$$

As before from each hypotheses a local neighborhood is used to find the maximum local score which is the hypothesis for the next resolution level. However, while in the rigid CtF algorithm, the local search was used to align the entire object model with the image features, now this procedure is done locally for each part, simulating local deformations.

In Fig. 4 a comparison of an object model learned with and without local deformations is shown. The model learned without local deformations is quite fuzzy, while the model learned with local deformations has stronger edges that make the model more discriminative.

3.3 Small Pedestrians

An important requirement in pedestrian detection for driving assistance is to detect low resolution pedestrian instances. Due to perspective distortion, low resolution pedestrians correspond to pedestrians far from the vehicle. Detecting far pedestrian gives enough time for a proper action to avoid collision, which is the first aim of a driving assistance system. However, when the number of pixels representing an object is low, the ability to recognize the object is highly reduced.

Concretely, for HOG computation, the number of pixels needed to build the features should always be the same to avoid under-sampling effects. This means that, when an object instance in an image has very low resolution, the corresponding HOG features can not be properly extracted and the object would be missed.

When using multiple resolutions, like in our case, a small instance object do not have fine resolution features, but it still has the coarse representation. So, in the CtF procedure, the search for the object can be extended to those scales that contains very small objects. In this case the missing high resolution features are set to zero. This allows the method to detect small objects. In this way, due to the missing contribution of the high resolution features, detections of small objects would have a score that is unbalanced (lower) with respect to full resolution detections because a part of the descriptor is artificially set to zero.

To overcome this problem, we extend [36] to our object model. We add to the feature descriptor a further binary feature for each resolution level, which represents whether, in the considered example, the corresponding resolution is available. Now the score is computed as:

$$D(\mathbf{x}) = \sum_{r=1:R} \sum_{p=1:P^{r-1}} \left\langle M_{r,p}, \left[H(\mathbf{x}_{r,p}), d_x^2, d_y^2, h_r\right]\right\rangle \quad (7)$$

where h_r is a binary variable that is enabled, when the corresponding HOG features $H(x_{r,p})$ are missing and therefore set to 0. In this way, h_r acts as a bias term that makes scores of detections generated without high resolution features comparable to full resolution detections. We evaluate the advantage of this solution in the experimental results section. For easy understanding in the rest of the paper we will refer to these additional features as "resolution" features.

From the computational point of view the increment of computation due to the use of the resolution feature is limited to the scan of the coarser resolutions of the model at the finer resolutions of the feature pyramid, which is actually much smaller than applying the detector to the whole image.

3.4 Learning

The learning procedure of our system is based on latent SVM [37], [21], [5]. Given a set of input data $\{x_1, \ldots, x_n\}$ and the associated labels $\{y_1, \ldots, y_n\}$, we find a parameter vector w of a function y that minimizes the regularized empirical risk:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i y(x_i, w)). \quad (8)$$

In our problem the input data x_i is the set of features extracted from the HOG pyramid H defined in the previous section and associated to an image region, while the output data y_i is a binary label indicating whether the object is present in the region. We introduce a latent variable $\mathbf{k}$ that represents the relative position of each child part with respect to its father. Considering the local position of each part allows the detector to learn a more discriminative model during learning and also to obtain a better alignment of the object model with the image data. The estimated output y is computed as:

$$\begin{aligned} y(x, w) &= \max_{\mathbf{k}} D_w(\mathbf{x} + \mathbf{k}) \\ &= \sum_{r=1:R} \sum_{p=1:P^{r-1}} \max_{\mathbf{k}_{r,p}} \left\langle M_{r,p}, \left[H(\mathbf{x}_{r,p} + \mathbf{k}_{r,p}), d_x^2, d_y^2, h_r\right]\right\rangle \end{aligned} \quad (9)$$

From Eq. (9) we see that w corresponds to the flattened and concatenated version of all the parts $M_{r,p}$ of our object model.

In contrast to normal SVM optimization, y is no longer linear in w due to the maximization on $\mathbf{k}$, therefore the empirical risk is no longer convex, and standard optimization techniques can not be used. Instead, we use the iterative procedure proposed in [5], where learning is divided into two iterative steps: the optimization of w with $\mathbf{k}$ fixed for the positive examples and the estimation of the best $\mathbf{k}$ using the computed w.

The optimization of w given $\mathbf{k}$ is convex and is computed using parallelized stochastic gradient descent [38]. The estimation of $\mathbf{k}$ with the current object model w is computed from Eq. (9). Instead of computing the exact maximization of Eq. (9), we apply the CtF procedure. Although there is not guarantee of the final performance of the approximate learning, we empirically see that it produces optimal results with a reduced computation.

While for the positive examples the object bounding box is given, for the negative examples a set of images not containing pedestrian is given (negative images).

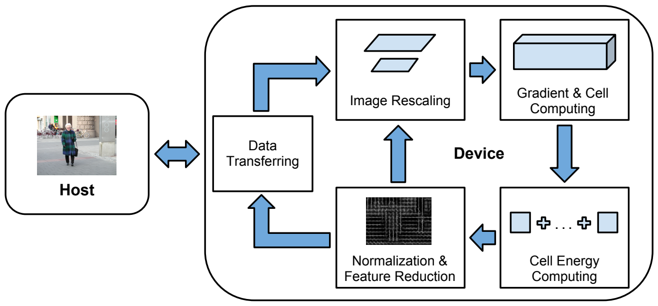

Fig. 5. GPU pipeline for the computation of the HOG features.

Thus, the negative examples at the beginning are selected as random bounding boxes from the negative images. Then, after a new model is built, the model itself is applied to the negative images in order to collect those regions of the image that are incorrectly classified as pedestrian and use them as extra negative examples.

4 Adapting the system for real-time computation

So far we have described a general detection system with a fast image scan, due to the Coarse-to-Fine procedure, which is able to detect small pedestrians thanks to the introduction of additional features that explicitly consider feature resolutions. However, as we will show in the experiments, although the system reaches a very high performance, still it is not fast enough for real-time application. In this sense, adapting the general framework to our specific task of pedestrian detection from a moving vehicle we can get some additional speed that can lead to real-time performance. In contrast to normal sliding window methods, where the main cost of a detection is produced by the image scan, in the coarse-to-fine procedure, the image scan is reduced by more than 10 times and therefore the dominant cost is the feature computation. In this regard, we propose to take advantage of the Graphics Processing Unit for the feature computation. Also, as we want to detect pedestrian in moving vehicles, although there is no stable background, we still know that the camera on the vehicle has a fixed inclination with the street. Assuming this fact, we avoid the evaluation of those image regions in which pedestrians are not likely to be.

4.1 Accelerating HOG computation by GPU

Graphic Processing Units (GPU) beside their general use in computer graphics, they can be also used for improving the speed of general algorithms using their high capability of parallel computing. By using CUDA, our algorithm designed in parallel manner can take advantages of the SIMT (Single-Instruction, Multiple-Threads) architecture, in which a block of threads can be executed concurrently on a streaming multiprocessor. Inspired by similar implementations [39], [30], we propose a fast HOG feature computation. Our implementation follows the design of [5]: in contrast to the standard HOG as defined in [20], [30], our implementation use both contrast insensitive and sensitive orientation channels, but substitutes the multiple normalizations of the HOG cells with additional normalization features. In this way, the final descriptor is smaller than the original HOG (31 dimensions instead of 36), but more discriminative.

The pipeline of HOG computation is divided into five steps (figure 5): image rescaling, gradient computation with spatial aggregation, energy summation in each cell, normalization with feature assembling and data transferring. After an image is transferred from host (CPU side) to device (GPU side), the other four steps will be executed in device iteratively until all scales are computed and results will be sent back to host. Generally, modern GPU can use hundreds of threads for each step on device, which produces an incredible acceleration, but many factors can delay the whole processing time.

In our case, the bottleneck is data transferring. It takes more than 50% of the time needed for the whole pipeline. It is easy to see that exchanging data with the host memory frequently might lose the time saved by parallel computing. Thus, an efficient way is to keep all steps executed in the GPU. In our implementation, we transfer an image from the global memory (off-chip memory) to GPU and keep it until the complete pyramid of HOGs is computed. For each scale, we resize the image, compute the gradient map and aggregate the gradient into the local histograms.

Since the final result of the HOG pyramid on GPU is exactly the same as in CPU, we can compare the two version just in the term of speed because the overall detection accuracy of the system will be the same. In figure 6 we show a comparison of HOG computation of a 40-level feature pyramid (an image downscale from 640×480 to 43×32, that is, 4 octaves and 10 intervals between 2 octaves) between GPU and CPU. We can see that GPU is in average 9.7 times faster than the CPU over all 40 scales.

4.2 Region of interest

In contrast to general object detection, pedestrian detection from a moving vehicle has some prior knowledge about the camera location and this can be used to further speed-up the final detector. In [13] for instance, the 3D location of the road (assuming it a plane) is estimated to reduce the pedestrian search only on this plane, therefore reducing the search cost. In contrast, we do not make any assumption about the road structure and do not estimate its 3D location. We take a simpler and conservative approach. Considering that the camera orientation in the vehicle is fixed and the maximum level of steep variation a road can present is limited, we explicitly avoid searching for pedestrians in the upper part of the

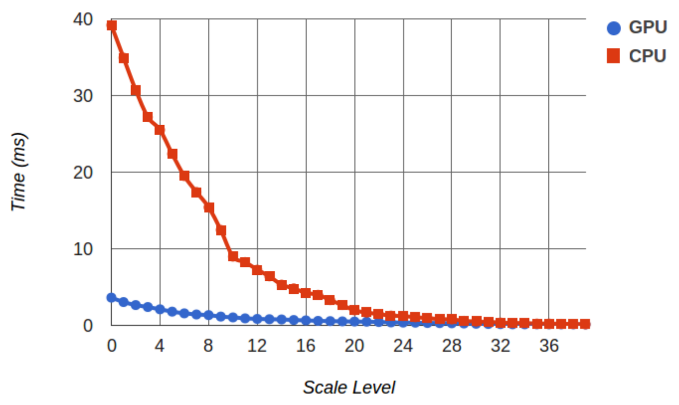

Fig. 6. Comparison of computation cost between CPU HOG and our GPU HOG.

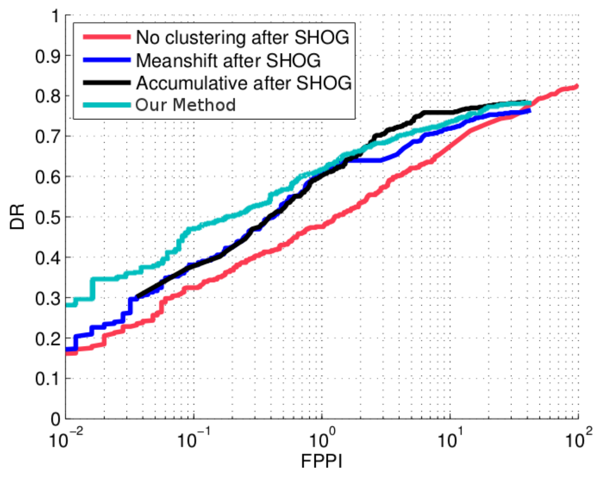

Fig. 7. Detection Rate versus False Positive Per Image on the CVC02 database. Our method corresponds to the CtF detector with deformations and “resolution” features activated (corresponding to the 7^{th} row of Table 1). The other curves are obtained from [40].

image: in our experiments, once the superior one third of the image is discarded, there is no loss in recall and a 30% saving of the total amount of computation.

5 Experiments

We evaluate our method on the CVC02 dataset [40], which is a dataset specific for pedestrian detection in the context of driving assistance. It consists of pedestrians taken in the range from 0 to 50 m, which correspond approximatively to 70×140 to 12×24 pixels bounding boxes. The training set consists of 1016 cropped humans with corresponding vertical mirror, for a total of 2032 images. The testing set consists of 250 urban images containing pedestrians. Examples of detections on test images of the CVC-02 dataset are shown in Fig. 8.

In Fig. 7 we compare our detector in terms of detection-rate (DR) versus false-positive-per-image (FPPI) with different configuration of the simplified HOG based on SVM learning detector (SHOG+SVM) proposed in [40]. For this experiment we use our CtF configuration with deformations and “resolution” features activated. Our detector has a quite relevant higher DR than the other when the working point is set to high precision (< 1 FPPI). The fastest configuration from [40] takes more than 10 s for detecting pedestrian in an image of the database (size 640×480 pixels). This is in line with our results using a complete image scan (see Table 1 rows 1 and 2). In contrast, our method on our machine (Intel Pentium Xeon 2.67 GHz using only one core) takes less than 1 s to compute the HOG pyramid on CPU and less than 0.1 s on GPU, whereas the image scan takes less than 0.5 s depending on the specific configuration.

5.1 Rigid versus Deformable models

In Table 1 we evaluate the quality of a detector configuration in terms of average precision (AP) and Time. AP is the averaged values of the precision obtained by the detector in a precision-recall curve drawn applying the detector the complete test set. *Time* is the average time needed for: (i) the image scan, i.e. searching the object in the image, (ii) the feature pre-compuation in both CPU and GPU and, (iii) the overall computation of a frame.

The first two rows of Table 1 compare a model using a rigid template (as explained in sect. 3.1) and a model using deformable parts (as explained in sect. 3.2). In both cases we use a complete search to avoid any possible problem due to the CtF scan. Learning local deformations through the object parts is useful to better align the object model with the image. This translate into an improved detector recall because also misaligned object can be correctly detected. In practice, the overall detector performance using the deformable model is increased of almost 5 points with respect to the rigid model with a relatively small increment of computational cost (from 8.48 to 10.73 seconds).

5.2 CtF versus Complete search

We next evaluate the performance of the CtF search compared with the complete search (i.e sliding window), both in terms of speed and accuracy. The first two rows of Table 1 have exactly the same configuration as row 3 and 4. The only difference is in the scan procedure. The first rows use complete search whereas the second CtF.

While the Average Precision (AP) of the two methods is comparable, CtF procedure scans an image in much less than 1 s while standard sliding windows takes up to 10 s. Note that for the rigid model (row 1 and 3) both complete and CtF searches give exactly the same AP. In case of deformable model, using the CtF approximation produces a loss in AP of 1 point. Still, the improvement compared to the rigid model is quite high, as well as the gain in time.

Image scan		Small windows		Image size		Model		Average	Time(s)				
Complete	CtF	Small	Feat	×1	×2	Rigid	Def	Precision(%)	Scan	Feat-CPU	Total-CPU	feat-GPU	Total-GPU
×				×		×		29.8	7.67	0.81	8.48	0.08	7.75
×				×			×	34.8	9.92	0.81	10.73	0.08	10.0
	×			×		×		29.8	0.14	0.81	0.95	0.08	0.22
	×			×			×	33.7	0.25	0.81	1.06	0.08	0.33
	×	×		×		×		59.4	0.24	0.81	1.05	0.08	0.32
	×	×		×			×	63.1	0.40	0.81	1.21	0.08	0.48
	×	×	×	×			×	65.0	0.40	0.81	1.21	0.08	0.48
	×	×			×	×		66.2	0.91	3.87	4.78	0.20	1.11
	×	×			×		×	73.1	1.36	3.87	5.23	0.20	1.56
	×	×	×		×		×	73.4	1.36	3.87	5.23	0.20	1.56

TABLE 1
Average precision (AP) and detection time with different configurations of our system. Image scan can be *Complete* if searching at all possible locations and correspond to standard sliding windows in case of using a rigid template, or *CtF* if using the faster procedure explained in section 3.1 for rigid templates and in section 3.2 for deformable templates. *small windows* correspond to evaluating also the windows where the high resolution model is not present without adding the corresponding feature *Small* or adding the corresponding "resolution" feature *Feat*. *image size* is the size of the image used for detection that can be the original $\times 1$ of 640×480 or the double $\times 2$. Finally the object *model* can be *rigid* or *deformable*.

5.3 Detection of Small Pedestrians

For pedestrian detection from a moving vehicle, the detection of small examples (far from the vehicle and therefore at low resolution) is fundamental. This is proved in the last part of Table 1. Here it is possible to notice the high AP improvement when adding the searching for objects at smaller size and the corresponding "resolution" features as explained in section 3.3. The AP for rigid model rise from 29.8 to 59.4. For deformable models the gap is almost similar: from 33.7 to 63.1 adding only the search of small objects and 65.0 adding also the "resolution" feature. In term of time, the search at a smaller resolution of the object adds some overhead in the image scan (from 0.14 to 0.24 for the rigid model and from 0.25 to 0.4 for the deformable). However, this is minor than the one introduced by doubling the image resolution. Doubling the image resolution would generates a slow-down of around $4\times$ in the image scan as well as in the feature computation. In contrast, the search of the object at smaller sizes is done only at low resolution, which is not very expensive but good enough to find many additional detections as confirmed by the improved AP.

Even though the average precision of the method is highly increased by using "resolution" features, the overall performance is still affected by small objects that in certain cases are missed. We evaluate this testing the method on resized images at double resolution. This configuration obtain the best AP gaining around 10% over the normal image size AP. However, the detection time per frame also highly increased, up to 5 s.

5.4 GPU for feature computation

From Table 1 and Fig. 6 we can see that the use of GPU has a stable speed-up in the feature computation of around 10 times. Compared with other implementations like [39], [30] this is not very high. However, we should remind that in contrast to other methods, in our current implementation, only the feature computation is computed on GPU, while the image scan is still done in CPU. This is mainly due to a design choice drawn from different reasons. First, implementing everything on GPU, and especially the recursive part of the CtF algorithm would be quite complex, long and prone to errors. Second, leaving the scan of the image in the CPU can be useful in case of further developments where multiple classes (i.e. cars, bicycles, etc...) should be detected at the same time. In this case, the algorithm can be easily extended to use multiple cores (nowadays quite common in standard PCs), each one for each class. Thus, the final detection speed would remain the same. This can not be exploited computing everything on GPU. Uue to the CtF procedure the time spent in the feature computation is much more relevant than in the complete scan. For instance, using the complete search with a rigid model, the CPU and GPU overall time for detection are respectively 8.48 and 7.45 s. The relative difference is not very relevant. In contrast, the total time in CPU for computing an image using CtF search is almost 1 second while for GPU is just 0.22 seconds, i.e. an overall speed-up of more than 4 times.

Also, as introduced in section 4.1, we can add a further improvement in speed by computing only the region of the image where pedestrians should appear. In the CVC02 dataset, using a selected region of the lower two thirds of the image produces a 33% speed-up while maintaining the same degree of accuracy. Note that all the reported times are computed using a single CPU. Taking advantage of multiple CPUs nowadays available in almost every PC could give a further boost.

Finally, using a 4 cores PC, the GPU computation of the features, and an additional reduction of the sampling

Method	Recall(at 10^{-1} FPPS)
Viola & Jones [1]	37%
Dalal & Triggs [20]	50%
fastHOG[39]	50%
groundHOG[30]	60%
OURS	74%
DPM[5]	75%

TABLE 2
Recall at 10^{-1} false positive per window (FPPI) on the INRIA pedestrian dataset.

scale in the pyramid computation leads to a complete detection system that is able to *run at around 10 frames per second*.

5.5 Comparison with other GPU-based detectors

A general comparison with other methods based on different implementations and different machines is quite difficult. However, one advantage of our work compared with previous pedestrian detectors is the use of deformations in conjunction with the CtF search, that give a high boost in performance and a small increment of computation. In this sense, we can compare our detector with those proposed in [39], [30], which are based on a fast GPU implementation of the Dalal and Triggs detector [20].

In Table 2 we compare those methods on the INRIA dataset [20]. The original pedestrian detector from Dalal & Triggs has a recall of 50% at 0.1 FPPI, which is the same as for the GPU implementation of fastHOG [39]. In contrast, probably due to a better HOG implementation, groundHOG reaches a recall of 60%. Our detector with deformable model at 0.1 FPPI has a recall of 74%. For completeness we also report the recall of the well known Viola and Jones [1] and the deformable part model [5].

6 Conclusions

In this paper we have presented a new framework for fast pedestrian detection in the context of driving assistance. The framework is based on the combination of recent state-of-the-art techniques for fast and accurate object detection in still images.

We have evaluated our system on a dataset specific for pedestrian detection from a moving vehicle and we have shown that it is able to outperform other fast detection methods in both speed and accuracy. This is due (i) the use of a CtF procedure for fast image scan, (ii) the use of obect parts to simulate local deformations (iii) the evaluation of detections with missing resolutions (iv) the introduction of an additional feature that balances out scores with missing resolutions and gives possibly high scores also to small detections, which are very important in the context of driving assistance.

Finally, we have proposed some techniques that exploiting domain-specific characteristics of pedestrian detection from a moving vehicle can further speed up the detection and lead to accurate and real-time performance.

Acknowledgments

Authors acknowledge the support of the Spanish Research Programs Consolider-Ingenio 2010: MIPRCV (CSD200700018); Avanza I+D ViCoMo (TSI-020400-2009-133) and Dicomo (TSI-020400-2009-133) along with the Spanish projects TIN2009-14501-C02-01 and TIN2009-14501-C02-02. Marco Pedersoli wants to acknowledge the Agency for Administration of University and Research Grants (AGAUR) of the Government of Catalonia for his scholarship;

References

[1] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *IJCV*, vol. 63, no. 2, pp. 153–161, 2005.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, june 2001.

[3] D. Gavrila and V. Philomin, "Real-time object detection for smart vehicles," in *ICCV*, october 1999, pp. 87–93.

[4] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *ICCV*, 2009.

[5] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *PAMI*, vol. 32, no. 9, 2010.

[6] D. Hoiem and M. H. Alexei A. Efros, "Putting objects in perspective," in *CVPR*, june 2006.

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

[8] C. G. Keller, M. Enzweiler, and D. M. Gavrila, "A new benchmark for stereo-based pedestrian detection," in *Intelligent Vehicles Symposium*, 2011, pp. 691–696.

[9] L. Zhao and C. Thorpe, "Stereo and neural network-based pedestrian detection," *IEEE Trans. on Intelligent Transportation Systems*, vol. 1, no. 3, pp. 148 –154, September 2000.

[10] D. M. Gavrila and S. Munder, "Multi-cue pedestrian detection and tracking from a moving vehicle," *Int. J. Comput. Vision*, vol. 73, no. 1, pp. 41–59, jun 2007. [Online]. Available: http://dx.doi.org/10.1007/s11263-006-9038-7

[11] W. van der Mark and D. M. Gavrila, "Real-time dense stereo for intelligent vehicles," *IEEE Trans. on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 38–50, 2006.

[12] S. Munder, C. Schnörr, and D. M. Gavrila, "Pedestrian detection and tracking using a mixture of view-based shape-texture models," *IEEE Trans. on Intelligent Transportation Systems*, vol. 9, no. 2, pp. 333–343, 2008.

[13] A. Sappa, F. Dornaika, D. Ponsa, D. Gernimo, and A. Lpez, "An efficient approach to onboard stereo vision system pose estimation," *IEEE Trans. on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 476–490, September 2008.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 2, no. 60, pp. 91–110, 2004.

[15] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proc. ICCV*, 2003.

[16] G. Csurka, C. R. Dance, L. Dan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. ECCV Workshop on Stat. Learn. in Comp. Vision*, 2004.

[17] S. Lazebnik and M. Raginsky, "Learning nearest-neighbor quantizers from labeled data by information loss minimization," in *Proc. Conf. on Artificial Intellligence and Statistics*, 2007.

[18] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *IJCV*, 2007.

[19] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," in *CVPR*, june 2011.

[20] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, p. 886893.

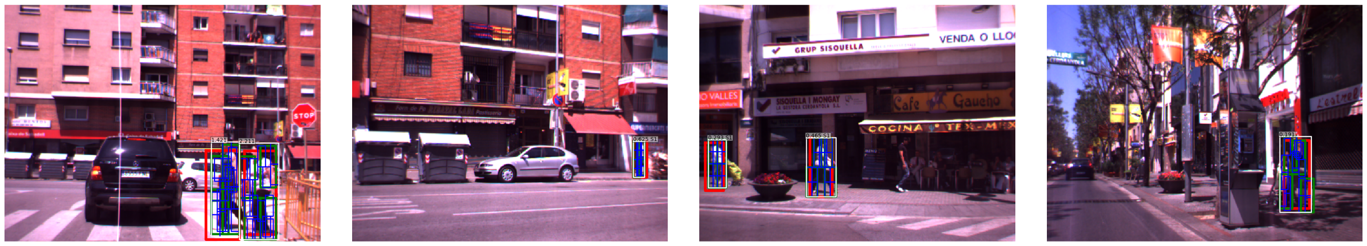

Fig. 8. Examples of detections from the test set of the CVC-02 dataset. For each image we show the bounding boxes of the detections with the corresponding score and the part configuration that led to the detection. When the detection score is followed by a S, it represents that the detection has been obtained using only the coarser resolutions (i.e. small object).

[21] A. Vedaldi and A. Zisserman, "Structured output regression for detection with partial occulsion," in *NIPS*, 2009.

[22] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, "Latent hierarchical structural learning for object detection," in *Proc. CVPR*, 2010.

[23] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Efficient subwindow search: A branch and bound framework for object localization," *PAMI*, vol. 31, no. 12, pp. 2129–2142, 2009.

[24] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *IJCV*, vol. 77, no. 1, pp. 259–289, May 2008.

[25] K. Mikolajczyk, B. Leibe, and B. Schiele, "Multiple object class detection with a generative model," in *CVPR*, june 2006, pp. 26–36.

[26] I. Laptev, "Improving object detection with boosted histograms," *Image and Vision Computing*, vol. 27, pp. 535–544, 2009.

[27] P. Dollar, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *Proc. BMVC*, 2010, pp. 68.1–11, doi:10.5244/C.24.68.

[28] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *CVPR*, 2012.

[29] P. F. Felzenszwalb, R. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *Proc. CVPR*, 2010.

[30] P. Sudowe and B. Leibe, "Efficient use of geometric constraints for sliding-window object detection in video," in *ICVS*, 2011.

[31] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *PAMI*, vol. 34, no. 4, pp. 743–761, 2012.

[32] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence, available online: IEEE Computer Society Digital Library, http://doi.ieeecomputersociety.org/10.1109/TPAMI.2008.260*, 2008.

[33] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *PAMI*, vol. 32, no. 7, pp. 1239–1258, 2010.

[34] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool, "The pascal visual obiect classes challenge 2007 (voc20067) results," ., Tech. Rep., 2007.

[35] M. Pedersoli, J. Gonzàlez, A. D. Bagdanov, and J. J. Villanueva, "Recursive coarse-to-fine localization for fast object detection," in *ECCV*, 2010.

[36] D. Park, D. Ramanan, and C. Fowlkes, "Multiresolution models for object detection," in *ECCV*, june 2010.

[37] C. N. J. Yu and T. Joachims, "Learning structural svms with latent variables," in *Proc. ICML*, 2009.

[38] M. Zinkevich, M. Weimer, A. Smola, and L. Li, "Parallelized stochastic gradient descent," in *Proc. NIPS*, 2010.

[39] V. Prisacariu and I. Reid, "fasthog - a real-time gpu implementation of hog," Department of Engineering Science, Oxford University, Tech. Rep. 2310/09, 2009.

[40] D. Gerónimo, "A global approach to vision-based pedestrian detetcion for advanced driver assistance systems," Ph.D. dissertation, Computer Vision Center, Universitat Autonoma de Barcelona, 2009.

Marco Pedersoli received the BSc degree from the Electronic Engineering Department, University of Brescia, Italy, in 2005 and the MSc degrees from the Computer Science Department, Autonomous University of Barcelona, Spain, in 2008. Since 2008 he is PhD student at the Computer Vision Center of Barcelona, Spain. His research is focused on localization and classification of visual object categories for image understanding.

Jordi Gonzàlez completed his PhD in Computer Engineering in 2004 at Universitat Autònoma de Barcelona (UAB). At present, he is Associate Professor in Computer Engineering at the Computer Science Department, UAB. He is also a research fellow at the Computer Vision Center. His research interests cover pattern recognition and machine learning techniques for the computational interpretation of human behaviours in image sequences, or Video Hermeneutics.

Xu Hu received his Bachelor's degree from the Department of Computer Science, Hangzhou Dianzi University, China, in 2010. He is pursuing his MSc degree in Computer Vision and Artificial Intelligence at Universitat Autònoma de Barcelona since 2011. His research involves object detection, pose estimation and action recognition.

Xavier Roca received his PhD in Computer Science in 1990 at Universitat Autònoma de Barcelona (UAB). At present, he is Associate Professor and Director of the Computer Science and Director Department, UAB. He is also a research fellow at the Computer Vision Center. He has been principal researcher in several projects (public and privates founds). He is working in technological transfer Computer Vision. The topic of his research are active vision and tracking.