Efficient Use of Geometric Constraints for Sliding-Window Object Detection in Video

Patrick Sudowe and Bastian Leibe

UMIC Research Centre
RWTH Aachen University, Germany

Abstract. We systematically investigate how geometric constraints can be used for efficient sliding-window object detection. Starting with a general characterization of the space of sliding-window locations that correspond to geometrically valid object detections, we derive a general algorithm for incorporating ground plane constraints directly into the detector computation. Our approach is indifferent to the choice of detection algorithm and can be applied in a wide range of scenarios. In particular, it allows to effortlessly combine multiple different detectors and to automatically compute regions-of-interest for each of them. We demonstrate its potential in a fast *CUDA* implementation of the *HOG* detector and show that our algorithm enables a factor 2-4 speed improvement on top of all other optimizations.

1 Introduction

Object detection has become a standard building block for many higher-level computer vision tasks. Current detectors reach sufficient detection accuracies [1] to support complex mobile scene analysis and multi-person tracking applications [2–4], and there is a strong call to make this performance available for automotive and robotics applications.

Even though first CPU [5] and GPU [6, 7] implementations of object detectors have already been proposed that operate at several frames per second, the pressure to develop more efficient algorithms does not subside. This is because object detection is only part of a modern vision system's processing pipeline and needs to share computational resources with other modules. In addition, practical applications often require not just detection of a single object category, but of many categories seen from multiple viewpoints [8]. Consequently, efficient object detection is a very active field of research. Many approaches have been proposed in recent years to speed up detection, including detection cascades [5, 9, 10], efficient approximative feature representations [11, 12], and alternatives to the sliding-window search strategy [13].

The use of scene geometry enables computational speedups which are orthogonal to the approaches mentioned above. It is well-known that in many street-scene scenarios, objects of interest can be expected to occur in a corridor of locations and scales on the ground plane [14, 3]. Approaches targeted at automotive scenarios have used such constraints for a long time. Surprisingly, though, the employed geometric constraints are often defined rather heuristically, sampling a few 3D locations and scales to be processed by the detector [15]. Such an approach is feasible for single-class detection scenarios with limited camera motion, but it quickly becomes impractical when multiple object class detectors shall be combined or when the camera may undergo stronger motion (*e.g.* for automatic sports video analysis or pan/tilt/zoom surveillance cameras).

In this paper, we derive a general solution for this problem that is applicable to *any camera*, *any ground plane*, and *any object detector or combination of detectors* (as long as perspective distortion is small enough such that objects can still be detected upright in the image). Starting from geometric principles, we analyze the space of sliding-window locations that correspond to geometrically valid object detections under the constraints of a ground plane corridor. We show that for a given detector scale, this space corresponds to an image region that is bounded by two parabolas, and we give a practical formula to efficiently compute the corresponding ROI. Based on this, we propose a sliding-window algorithm that touches the minimal set of pixels to return all valid detections.

Our approach is flexible. It does not rely on a precomputed ground plane corridor, but provides a principled algorithm to *recompute the ROI for every frame* based on an estimate of the camera motion. The only information it requires is the current ground plane homography, the projection of the ground plane normal vector (both of which can be obtained either by structure-from-motion [3] or homography tracking [16]), the height of the detection bounding box in the image, and the real-world size range of the objects of interest.

In particular, this makes it possible to combine multiple object detectors with minimum effort. It does not matter whether those detectors have been trained on different resolutions [17] or for different viewpoints or real-world object sizes [18]. Everything that is needed is each detector's bounding box height in the image and the target object's real-world size range. We demonstrate this by performing experiments for single-class pedestrian detection [19] and for multi-viewpoint car detection (similar to [3, 18]). In all cases, we show the validity of our approach and quantify the resulting detector speed-ups. In order to perform a fair quantitative evaluation of those speed-ups, it is important to apply our algorithm to an already efficient detector implementation. We therefore combine it with a fast CUDA implementation of HOG, which closely follows the original HOG pipeline from [19]. Our resulting *groundHOG* pedestrian detector runs at 57*fps* for a street scene scenario without loss in detection accuracy.

Related Work Several recent approaches have been proposed to integrate scene geometry and detection [14, 4, 20–22]. Their main goal is to increase precision by selecting consistent detections. However, scene geometry offers additionally a potential speed increase, if one limits the detector's search region. Many automotive applications therefore employ a fixed, precomputed ground plane corridor (*e.g.* [2]), which is deliberately left a bit wider than necessary in order to compensate for changing camera pitch. Other approaches try to stabilize the camera image by detecting the horizon line [23] or fit a ground plane to stereo measurements [15]. A common approach is to then sample a fixed set of ROIs in 3D and to process the corresponding image regions with an AdaBoost classifier [15]. Such an approach is possible when dealing with a single object category, but it quickly becomes both inefficient and cumbersome when multiple categories with different real-world sizes shall be detected simultaneously. Such a scenario requires a more principled solution.

From the computational side, there are two major cost items in the design of a sliding-window classifier: the evaluation of the window classifier itself and the computation of the underlying feature representation. The success of the Viola-Jones detector [5] has shown that for certain object classes such as faces or cars, relatively simple Haar wavelet features are sufficient. This has been used in the design of AdaBoost based detectors which evaluate the features for each test window independently. For more complex object categories, Histograms of Oriented Gradients (HOG) [19] have become the dominant feature representation [1]. Unfortunately, HOG features are expensive to compute. Looking at highly optimized CUDA implementations of the HOG detection pipeline [6, 7], they typically account for 60-70% of the total run-time of a single-class classifier. It is therefore more efficient to precompute the features and to reuse them for different evaluation windows [24, 25], as is common practice in the design of SVM-based detectors [19, 18]. When combining several classifiers for different object aspects or categories, the relative importance of the shared feature computation decreases, but it still imposes a lower bound on the effective run-time.

In this paper, we therefore address both problems together: (1) Our proposed algorithm automatically computes the ROIs in which each employed classifier needs to be evaluated for each detection scale, such that it only considers geometrically valid detections. (2) In addition, it returns the minimal set of pixels that need to be touched for all detectors together, so that feature computation can be kept efficient.

The paper is structured as follows. We first derive a general formulation for the problem and analyze the space of sliding-window locations that correspond to geometrically valid object detections (Sec. 2). Based on this analysis, we propose a general algorithm for incorporating ground plane constraints directly into the detector design (Sec. 3). Finally, Sec. 4 presents detailed experimental results evaluating the approach's performance in practice.

2 The Space of Valid Object Detections

The central question we address in this paper is: What is the space of all valid detections? That is, if we only consider detection bounding boxes that correspond to objects on the ground plane whose real-world size is within a range of $S_{obj} \in [S_{min}, S_{max}]$, what is the region in the image in which those bounding boxes can occur? More concretely, we consider a sliding-window detector that processes the image at a discrete set of scale levels. At each scale, a fixed-size bounding box of height s_{img} pixels slides over the image. We are interested in the positions of the bounding box foot point y_b that lead to valid detections.

Geometric Derivation. In the following, we address this problem in the general case. We use the notation from [26], denoting real-world quantities by upper-case letters and image quantities by lower-case letters. Let us assume that we have a calibrated camera with projection matrix $\mathtt{P} = \mathtt{K}\,[\mathtt{R}|\mathbf{t}]$ watching a scene containing the ground plane $\boldsymbol{\pi}$ with normal vector $\mathbf{N}$(Fig. 1). We can define a local coordinate system on the ground plane by an origin $\mathbf{Q}_0$ and two orthogonal basis vectors $\mathbf{Q}_1, \mathbf{Q}_2$. The (homogenous) world coordinates $\mathbf{X} = [X, Y, Z, 1]^\mathsf{T}$ of a point $\mathbf{U} = [U, V, 1]^\mathsf{T}$ on the ground plane are then given by the transformation

$$\mathbf{X} = \mathtt{Q}\mathbf{U} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 & \mathbf{Q}_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{U} \tag{1}$$

and their projection on the image plane is given by the homography $\mathtt{H}_{\boldsymbol{\pi}} = \mathtt{PQ}$.

We now want to find an object with real-world height S_{obj} that is located on or above ground plane position $\mathbf{U}$ and which extends from height S_b to height $S_t = S_b + S_{obj}$. The projections $\mathbf{x} = [x, y, w]^\mathsf{T}$ of the object's bottom and top points $\mathbf{X}_b$ and $\mathbf{X}_t$ into the image are given by

$$\mathbf{x}_b = \mathtt{P}\mathbf{X}_b = \mathtt{P}\left(\mathtt{Q}\mathbf{U} + S_b\mathbf{N}\right) = \mathtt{H}_{\boldsymbol{\pi}}\mathbf{U} + S_b\mathrm{P}\mathbf{N} \tag{2}$$

$$\mathbf{x}_t = \mathtt{P}\mathbf{X}_t = \mathtt{P}\left(\mathtt{Q}\mathbf{U} + S_t\mathbf{N}\right) = \mathtt{H}_{\boldsymbol{\pi}}\mathbf{U} + S_t\mathrm{P}\mathbf{N}. \tag{3}$$

Writing $\mathbf{h}_j^\mathsf{T} = [h_{j1}, h_{j2}, h_{j3}]$ for the row vectors of $\mathtt{H}_{\boldsymbol{\pi}}$ and using $\mathbf{n} = [n_1, n_2, n_3]^\mathsf{T} = \mathtt{P}\mathbf{N}$, we can compute the y coordinates of the corresponding image pixels as

$$y_b = \frac{\mathbf{h}_2^\mathsf{T}\mathbf{U} + S_b n_2}{\mathbf{h}_3^\mathsf{T}\mathbf{U} + S_b n_3}\ , \qquad y_t = \frac{\mathbf{h}_2^\mathsf{T}\mathbf{U} + S_t n_2}{\mathbf{h}_3^\mathsf{T}\mathbf{U} + S_t n_3}\ . \tag{4}$$

We can now express the constraint that the projected object height in the image should exactly correspond to the height of the sliding window given by s_{img}:

$$y_t = y_b + s_{img} \tag{5}$$

$$\frac{\mathbf{h}_2^\mathsf{T}\mathbf{U} + S_t n_2}{\mathbf{h}_3^\mathsf{T}\mathbf{U} + S_t n_3} = \frac{\mathbf{h}_2^\mathsf{T}\mathbf{U} + S_b n_2 + s_{img}\left(\mathbf{h}_3^\mathsf{T}\mathbf{U} + S_b n_3\right)}{\mathbf{h}_3^\mathsf{T}\mathbf{U} + S_b n_3}$$

$$\left(\mathbf{h}_2^\mathsf{T}\mathbf{U}+S_t n_2\right)\left(\mathbf{h}_3^\mathsf{T}\mathbf{U}+S_b n_3\right) = \left(\mathbf{h}_2^\mathsf{T}\mathbf{U}+S_b n_2+s_{img}\left(\mathbf{h}_3^\mathsf{T}\mathbf{U}+S_b n_3\right)\right)\left(\mathbf{h}_3^\mathsf{T}\mathbf{U}+S_t n_3\right)$$

The set of all ground plane locations $\mathbf{U}$ for which this constrained is fulfilled is then given by the conic section $\mathtt{C}$ with

$$\mathbf{U}^\mathsf{T}\mathtt{C}\mathbf{U} = 0 \tag{6}$$

$$\begin{bmatrix} U & V & 1 \end{bmatrix} \left[\mathbf{h}_3\mathbf{h}_3^\mathsf{T} + \begin{bmatrix} 0 & 0 & a \\ 0 & 0 & b \\ a & b & c+d \end{bmatrix} \right] \begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = 0\ , \quad \text{where}$$

$$\begin{bmatrix} 2a & 2b & c \end{bmatrix} = \frac{1}{s_{img}} \left[(S_t - S_b)(n_3\mathbf{h}_2^\mathsf{T} - n_2\mathbf{h}_3^\mathsf{T}) + s_{img}(S_t + S_b)n_3\mathbf{h}_3^\mathsf{T} \right] \tag{7}$$

$$d = S_t S_b n_3^2\ . \tag{8}$$

It can easily be seen that the discriminant of the conic (*i.e.*, the determinant of its upper-left 2×2 matrix) is 0, since $\mathbf{h}_3\mathbf{h}_3^\mathsf{T}$ has only rank 1. The equation therefore represents a parabola, whose projection into the image is given by

$$\mathbf{x}^\mathsf{T}\mathtt{D}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathtt{H}_\pi^{-\mathsf{T}}\mathtt{C}\mathtt{H}_\pi^{-1}\mathbf{x} = 0 \ . \tag{9}$$

Analysis. In our sliding-window detection scenario, we are interested in finding objects which have a real-world height in the range $S_{obj} \in [S_{min}, S_{max}]$. From the above derivation, it follows that the only windows at which those objects can be found are located in the space between the two curves defined by $\mathtt{D}$ for $S_t = S_b + S_{min}$ and $S_t = S_b + S_{max}$. In the following, we analyze the detailed shape of those curves further. If the camera viewing direction is exactly parallel to the ground plane, then eq. (9) degenerates and defines a pair of lines (one of which will be behind the camera). In order to analyze the remaining cases, we perform the variable substitution

$$\begin{bmatrix}\bar{U}\\ \bar{V}\end{bmatrix} = \begin{bmatrix} h_{31} & h_{32}\\ -h_{32} & h_{31}\end{bmatrix}\begin{bmatrix}U\\ V\end{bmatrix} \tag{10}$$

and obtain the ground plane locations of the curve points on the parabola

$$\bar{V} = \frac{h_{31}^2 + h_{32}^2}{2(h_{32}a - h_{31}b)}\bar{U}^2 + \frac{h_{33}(h_{31}^2 + h_{32}^2) + h_{31}a + h_{32}b}{h_{32}a - h_{31}b}\bar{U} + \frac{(h_{33}^2 + c + d)(h_{31}^2 + h_{32}^2)}{2(h_{32}a - h_{31}b)} \ .$$

Of particular interest is the factor in front of the quadratic term $\bar{U}^2$. In a more detailed analysis we found that the factor is negligible for most practically relevant cases in automotive or mobile robotics scenarios, unless a wide-angle camera is used. This means the parabola can be approximated by a line.

Obtaining the Ground Constraints. In the above derivation, we assumed an internally and externally calibrated camera, as well as knowledge about the ground plane. In an automotive or mobile robotics setup, this information can be obtained by structure-from-motion and dense stereo measurements (*e.g.* [3, 4]). However, looking at the components of $\mathtt{D}$ in eq. (9), it becomes clear that the curve is already fully specified if we know the ground plane homography $\mathtt{H}_\pi$ and the projection of the normal vector $\mathbf{n} = \mathtt{P}\mathbf{N}$. This makes the approach also attractive for other applications, such as sports broadcasts or surveillance, where landmark points on the ground plane can be tracked to maintain calibration.

The homography $\mathtt{H}_\pi$ can be estimated from at least four image points with known ground plane coordinates (*e.g.* using the DLT algorithm [26]). The projection of the normal can also easily be obtained from two or more points with known heights above the ground plane. Let $\mathbf{X}_i$ be a point with height S_i above its known ground plane footpoint $\mathbf{U}_i$. According to eq. (4), the corresponding image coordinates are given by

$$x_i = \frac{\mathbf{h}_1^\mathsf{T}\mathbf{U}_i + S_i n_1}{\mathbf{h}_3^\mathsf{T}\mathbf{U}_i + S_i n_3} \ , \qquad y_i = \frac{\mathbf{h}_2^\mathsf{T}\mathbf{U}_i + S_i n_2}{\mathbf{h}_3^\mathsf{T}\mathbf{U}_i + S_i n_3} \ . \tag{11}$$

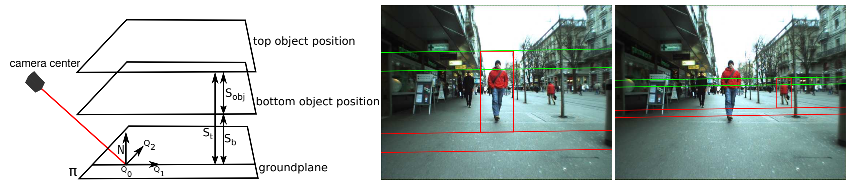

Fig. 1. (left) Visualization of the employed coordinate system and notation. (middle & right) Ground plane corridor at scales $\sigma = 1.75$ (middle) and $\sigma = 0.65$ (right). Two valid detections within the corridor are shown. The selected region-of-interest (ROI) is delimited by the uppermost and lowermost lines.

From this, we get an equation system, with two constraints per measured point:

$$\begin{bmatrix} -S_i & 0 & S_i x_i \\ 0 & -S_i & S_i y_i \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} = \begin{bmatrix} (\mathbf{h}_1^\top - x_i \mathbf{h}_3^\top) U_i \\ (\mathbf{h}_2^\top - y_i \mathbf{h}_3^\top) U_i \\ \vdots \end{bmatrix} \tag{12}$$

$$\mathtt{A}\mathbf{n} = \mathbf{b} \ , \tag{13}$$

resulting in the least-squares solution $\mathbf{n} = \mathtt{A}^\dagger \mathbf{b}$ if at least two points are given.

Extension to Multiple Scales. Until now, we have assumed that the image is scanned with a fixed-size detection window with height s_{img}. In order to detect objects at different scales, a sliding-window detector processes downscaled versions of the input image at fixed scale intervals σ. We achieve the same effect by adapting the internal camera calibration matrix $\mathtt{K}$. If we assume a camera with zero skew, this results in the following matrix for scale level k:

$$\mathtt{K}_k = \begin{bmatrix} \alpha_x/\sigma^k & 0 & x_0/\sigma^k \\ 0 & \alpha_y/\sigma^k & y_0/\sigma^k \\ 0 & 0 & 1 \end{bmatrix} \ . \tag{14}$$

Propagating this change to the ground plane homography and the projection of the normal vector, we can see that those entities are obtained as

$$\mathtt{H}_{\boldsymbol{\pi},k} = \begin{bmatrix} \mathbf{h}_1^\top/\sigma^k \\ \mathbf{h}_2^\top/\sigma^k \\ \mathbf{h}_3^\top \end{bmatrix} \ , \qquad \mathbf{n}_k = \begin{bmatrix} n_1/\sigma^k \\ n_2/\sigma^k \\ n_3 \end{bmatrix} \ . \tag{15}$$

3 Detection Algorithms

Putting all the pieces together, we can now formulate a general algorithm for geometrically constrained object detection, as shown in Alg. 1. For each scale level, we first compute the corresponding $\mathtt{D}$ matrices for the minimum and maximum object size. We then create a rectangular ROI by inserting the x coordinates of the left and right image borders into eq. (9) and taking the minimum and maximum of the resulting y coordinates. As derived above, only the window locations inside this region correspond to geometrically valid object detections. Since we

Algorithm 1 The proposed algorithm

```
Compute $\mathtt{H}_{\boldsymbol{\pi}}$ and $\mathbf{n}$.
for all scale levels $k$ do
    Compute $\mathtt{H}_{\boldsymbol{\pi},k}$ and $\mathbf{n}_k$ according to eq. (15).
    Compute $\mathtt{D}_{S_{min}}$ and $\mathtt{D}_{S_{max}}$ according to eq. (9) using $\mathtt{H}_{\boldsymbol{\pi},k}$.
    Set $x_{min}$ and $x_{max}$ to the left and right image borders.
    Compute $y_{min}$ and $y_{max}$ by solving eq. (9) for $x_{min}$ and $x_{max}$ using $\mathtt{D}_{S_{min}}$ and $\mathtt{D}_{S_{max}}$.
    Process the ROI $(x_{min}, y_{min}, x_{max}, y_{max})$ with the detector:
    • Only up-/downscale the image pixels inside the ROI.
    • Only compute features inside the ROI.
    • Only apply the sliding-window classifier to the window locations in the ROI.
end for
```

compute the region for each scale independently, this allows us to *restrict all rescaling and feature computation steps* to those regions.

Multi-Class/Multi-viewpoint Detection. A straightforward extension to multiple classes or viewpoints of objects is to apply several specialized classifiers on the precomputed features. This approach can be easily augmented with our geometric constraints formulation. For each individual classifier one precomputes the *ROI.* The *HOG* features are then computed for a minimal region encompassing all *ROIs* that are active at each scale. Each classifier can then be evaluated on its respective region. Note that only the height of each object class and the classifiers' window sizes are necessary. No error-prone manual process is required.

Different Detector Resolutions and Part-based Models. A common method to improve detection performance is to use specialized classifiers for distinct scale ranges [10]. Our formulation naturally adapts to the *ROI* multi-resolution case. Here, the benefit is that the system can determine automatically if at some scale only a subset of classifiers can return viable detections. It is *not necessary* to fine tune any further parameters. If no valid *ROI* is found for a classifier, it is automatically skipped for the current scale. Similarly, our algorithm can be used with the popular part-based detection approach by Felzenszwalb *et al.* [18].

4 Experimental Results

We quantitatively evaluate our proposed ground plane constraints. In order to demonstrate the advantage our algorithm can achieve *on top of all other optimizations*, we combine it with a highly optimized *CUDA* implementation of the *HOG* detector (in the following called *cudaHOG*). Our code is publicy available at `http://www.mmp.rwth-aachen.de/projects/groundhog`.

Baseline Detection Performance. First, we establish that our baseline system *cudaHOG* achieves the same detection performance as two other published HOG-based systems: the original Dalal HOG Detector [19] and fastHOG [7]. Fig. 3(left) compares the performance on the INRIA pedestrian dataset [19]. We plot recall *vs.* false positives per image (fppi) using the standard PASCAL VOC criterion [27]. The plot shows that our baseline *cudaHOG* implementation is competitive.

Effect of Ground Plane Constraints. Next, we investigate the effects of the ground plane constraints in detail. For the experiments in this section, we

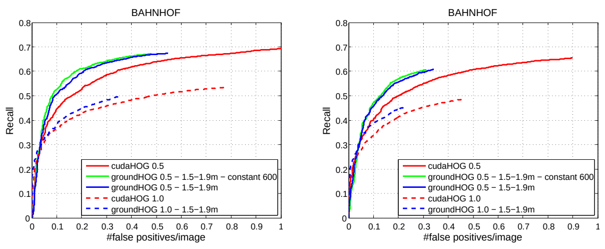

Fig. 2. *cudaHOG vs. groundHOG* for scale steps 1.05 (left) and 1.20 (right). In both cases, we plot the performances when starting at scale 1.0 and when upscaling the image to twice its original resolution (scale 0.5). For the upscaled version we also plot the performance for a bounded ROI width of maximal 600 pixels.

use the BAHNHOF sequence from the Zurich Mobile Pedestrian corpus [4], and employ ground planes estimated by SfM. The sequence consists of 999 frames of size 640×480, containing 5,193 annotated pedestrians with a height > 60 pixels.

We start by evaluating computational effort on the first 100 frames. We vary the start scale and ground plane corridor size and report the number of blocks and SVM windows evaluated, as well as the average run-time per frame (Tab. 1). Our baseline *cudaHOG* runs at roughly 22 fps for the start scale 1.0. By adopting a ground plane corridor of $[1.5m, 1.9m]$, we can more than double the speed to 57 fps.

In addition, we investigate how detection performance is affected by the start scale. As observed by several authors [1, 3, 4], the HOG detection performance can be considerably improved by upscaling the input images to twice their original resolution (start scale $\sigma = 0.5$ instead of $\sigma = 1.0$). The results shown in Fig. 2 verify this performance improvement (*e.g.*, recall increases by 10% at 0.2 fppi). Usually, the upscaling step comes at considerable additional cost. *groundHOG* can achieve significant computational savings here, since it can limit the upscaling operation to a (relatively small) band around the horizon line. As Tab. 1 shows, *groundHOG* can still process the upscaled images at 20 fps (23 fps if the width of the detection corridor is also bounded to 600 pixels), effectively the same run-time as the unconstrained detector on the original images. Hence, our algorithm achieves *significantly higher recall* in the *same computation time*.

Finally, we investigate the effect of increasing the scale step factor σ from its default value of 1.05 to 1.20 (as also explored in [6]). As shown in Tab. 1 and Fig. 2, this results in a significant speedup to 222 fps without and 87 fps/104 fps with upscaling at a moderate loss of recall (about 5% at 0.5 fppi).

Multi-Class/Multi-viewpoint Detection. As a proof-of-concept experiment for multi-viewpoint detection, we have trained a basic car detector for five viewpoints. We perform a bounding box based non-maximum-suppression step on the individual detector outputs to combine them into a single detection response. While this basic setup cannot achieve the absolute detection rates of more sophisticated setups, it is suitable to demonstrate the effects of a ground plane constraint. We evaluate on the LEUVEN sequence [3] that contains 1175 images

	scale step 1.05					scale step 1.2				
	start 1.0		start 0.5			start 1.0		start 0.5		
	cuda	ground	cuda	ground	max w	cuda	ground	cuda	ground	max w
HOG blocks	53,714	21,668	215,166	52,230	40,781	16,305	6,334	65,404	15,142	11,460
SVM windows	31,312	4,069	162,318	11,132	8,208	9,801	1,243	50,110	3,289	2,341
run-time (ms)	43.78	17.28	183.60	49.80	43.49	12.44	4.50	50.35	11.45	9.58
run-time (fps)	22	57	5	20	23	80	222	19	87	104

Table 1. HOG blocks & SVM windows evaluated per frame on the BAHNHOF sequence when applying *groundHOG* with the corridor $[S_{min}, S_{max}] = [1.5m, 1.9m]$. max w refers to a maximal ROI width of 600 pixels. (CPU: Core2Quad Q9550, GPU: GTX 280)

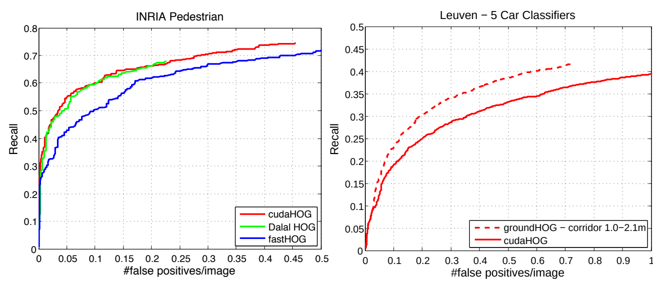

Fig. 3. (left) Baseline comparison on *INRIAPerson* dataset. (right) Results of a 5-view car detector on LEUVEN sequence, demonstrating the performance gains through our geometric constraints. The individual results are merged by a simple NMS scheme.

at 720 × 576 pixel resolution. Fig. 3 shows that detection performance benefits significantly, as fewer false positives are encountered by *groundHOG*. When incorporating the ground plane constraints, detection takes only 94 ms compared to originally 339 ms, representing a 3.6-fold speedup.

5 Conclusion

We have systematically explored how geometric ground plane constraints can be used to speed up sliding-window object detection. As a result of this analysis, we have presented a general algorithm that enforces a detection corridor, while taking maximum advantage of the sliding window detection scheme. We have demonstrated this approach in a CUDA implementation of the *HOG* detector. As verified in our experiments, the resulting *groundHOG* algorithm achieves at least the same detection accuracy as the original *HOG* detector in a range of detection scenarios, while allowing significant speedups.

Acknowledgments. This project has been funded, in parts, by the EU project EUROPA (ICT-2008-231888) and the cluster of excellence UMIC (DFG EXC 89).

References

1. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian Detection: A Benchmark. In: CVPR. (2009)
2. Gavrila, D., Munder, S.: Multi-Cue Pedestrian Detection and Tracking from a Moving Vehicle. IJCV **73**(1) (2007) 41–59

3. Leibe, B., Schindler, K., Van Gool, L.: Coupled Object Detection and Tracking from Static Cameras and Moving Vehicles. PAMI **30**(10) (2008) 1683–1698
4. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: Robust Multi-Person Tracking from a Mobile Platform. PAMI **31**(10) (2009) 1831–1846
5. Viola, P., Jones, M.: Robust Real-Time Face Detection. IJCV **57**(2) (2004)
6. Wojek, C., Dorko, G., Schulz, A., Schiele, B.: Sliding Windows for Rapid Object Class Localization: A Parallel Technique. In: DAGM. (2008)
7. Prisacariu, V., Reid, I.: fastHOG – a Real-Time GPU Implementation of HOG. Technical Report 2310/09, Dept. of Eng. Sc., Univ. of Oxford (2009)
8. Torralba, A., Murphy, K., Freeman, W.: Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection. In: CVPR. (2004)
9. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple kernels for object detection. In: ICCV. (2009)
10. Felzenszwalb, P., Girshick, R., McAllester, D.: Cascade Object Detection with Deformable Part Models. In: CVPR. (2010)
11. Dollar, P., Tu, Z., Perona, P., Belongie, S.: Integral Channel Features. In: BMVC. (2009)
12. Dollar, P., Belongie, S., Perona, P.: The Fastest Pedestrian Detector in the West. In: BMVC. (2010)
13. Lampert, C., Blaschko, M., Hofmann, T.: Efficient Subwindow Search: A Branch and Bound Framework for Object Localization. PAMI **31**(12) (2009) 2129–2142
14. Hoiem, D., Efros, A., Hebert, M.: Putting Objects Into Perspective. In: CVPR. (2006)
15. Geronimo, D., Sappa, A., Ponsa, D., Lopez, A.: 2D-3D-based On-Board Pedestrian Detection System. CVIU **114**(5) (2010) 583–595
16. Choi, W., Savarese, S.: Multiple Target Tracking in World Coordinate with Single, Minimally Calibrated Camera. In: ECCV. (2010)
17. Park, D., Ramanan, D., Fowlkes, C.: Multiresolution models for object detection. In: ECCV. (2010)
18. Felzenszwalb, P., McAllester, D., Ramanan, D.: A Discriminatively Trained, Multiscale, Deformable Part Model. In: CVPR. (2008)
19. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: CVPR. (2005)
20. Breitenstein, M., Sommerlade, E., Leibe, B., Van Gool, L., Reid, I.: Probabilistic Parameter Selection for Learning Scene Structure from Video. In: BMVC. (2008)
21. Bao, Y., Sun, M., Savarese, S.: Toward Coherent Object Detection And Scene Layout Understanding. In: CVPR. (2010)
22. Sun, M., Bao, Y., Savarese, S.: Object Detection with Geometrical Context Feedback Loop. In: BMVC. (2010)
23. Bombini, L., Cerri, P., Grisleri, P., Scaffardi, S., Zani, P.: An Evaluation of Monocular Image Stabilization Algorithms for Automotive Applications. In: Intel. Transp. Syst. (2006)
24. Schneiderman, H.: Feature-Centric Evaluation for Efficient Cascaded Object Detection. In: CVPR. (2004)
25. Lehmann, A., Leibe, B., Van Gool, L.: Feature-Centric Efficient Subwindow Search. In: ICCV. (2009)
26. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge University Press (2000)
27. Everingham, M., others (34 authors): The 2005 PASCAL Visual Object Class Challenge. LNAI 3944. Springer (2006)