Fast Stixel Computation for Fast Pedestrian Detection

Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool

ESAT-PSI-VISICS/IBBT, KU Leuven, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. Applications using pedestrian detection in street scene require both high speed and quality. Maximal speed is reached when exploiting the geometric information provided by stereo cameras. Yet, extracting useful information at speeds higher than 100 Hz is a non-trivial task. We propose a method to estimate the ground-obstacles boundary (and its distance), without computing a depth map. By properly parametrizing the search space in the image plane we improve the algorithmic performance, and reach speeds of 200 Hz on a desktop CPU. When connected with a state of the art GPU objects detector, we reach high quality detections at the record speed of 165 Hz.

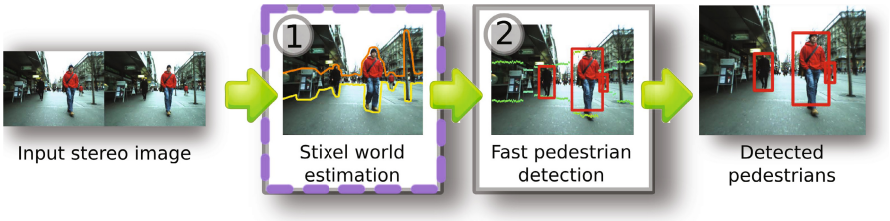


Fig. 1. Fast pedestrian detection pipeline (based on [1]). This paper focuses on speeding up the stixel world estimation (step 1) via algorithmic improvements.

1 Introduction

Fast object detection is of utmost importance in ground mobile robots. The faster the objects are detected and categorized, the faster the robot will react to them, enabling high speed displacement and/or smooth natural motion. False positive detections should be minimized, to avoid unnatural motions; false negative detections should be minimized for safety reasons. On the other hand, the computational budget is restricted due to constrains on power consumption and the need to execute other modules on the same platform.

This tension between speed and quality has motivated a significant amount of research. Recently a new approach for fast pedestrian detection has been proposed [1]. The authors proposed a method for high quality detections at 50 Hz

A. Fusiello et al. (Eds.): ECCV 2012 Ws/Demos, Part III, LNCS 7585, pp. 11–20, 2012.

in the monocular case (using GPU), and proposed to exploit the geometrical information to reach 135 Hz in the stereo case (using GPU + CPU). To reach such high speed in the stereo case, Benenson et al. proposed to skip the usual depth map computations stage, and instead use a direct method to estimate the presence of objects above the ground [2] ("stixel world" estimation [3]). Combining such fast detection method with fast stereo image processing, allows to detect pedestrians in less that 10 milliseconds per image.

In [1] the authors report than the current method is CPU-bound; although GPU detection runs at $\sim$ 150 Hz, the stereo processing only reaches 135 Hz, becoming the limiting factor. In this paper we revisit the stereo processing method, and show that with a proper re-parametrization, it is possible to reach 200 Hz for the stixel world estimation. This new method enables reaching high quality detections at 165 Hz, and frees CPU resources for additional tasks (e.g. tracking, planning).

1.1 Related Work

The idea of exploiting stereo information to speed-up objects detection has been around since more than fifteen years [4] . Multiple methods for coupling depth maps and objects detection have been proposed [5,6,7,8,9,10,11]. Similarly, free space estimation allows to disregard areas of the image where we know obstacles are not present [12,13].

The vast majority of methods previously proposed compute as a first step a dense (or semi-dense) depth map of the scene, and then use this depth map to infer the presence of objects. When doing this, the depth map computation becomes the speed bottleneck, since it needs to compute much more information than needed ("distance of every pixel in the image" versus "where are the objects?"). Badino et al. introduced the notion of "stixel world", which can be seen as the minimal world model useful to describe the objects, it simply includes a ground plane and objects that "stick out" from the ground [3,2].

Kubota et al. [14] showed that depth maps could be skipped, and the distance to objects could be directly estimated. Benenson et al. [2] then extended this work by showing that height estimation could also be done without depth map computation, and that the stixel world estimated by such an approach is suitable for objects detection [1,15] and for object motion estimation [16].

1.2 Contribution

Although direct stixel world estimation [2] provides a significant speed-up (from $\sim$ 20 Hz to $\sim$ 100 Hz), when coupled with state of the art GPU objects detector, it lags behind becoming the speed bottleneck [1]. This is the problem we address in this paper, our **key contribution** is proposing a new parametrization for direct stixel world computation (without depth map) which allows to reach $\sim$ 200 Hz on CPU without compromising on the detection quality.

All together we reach high quality detections at 165 Hz (CPU+GPU), with the GPU becoming the bottleneck, and freeing CPU resources for other tasks.

In section 2 we describe our stixel world estimation method. Section 3 explains how the stixels are used to accelerate and improve objects detection. Section 4 provides empirical evidence of the improved quality versus speed trade-off, both for stixels computation per se and when coupled with objects detection. We conclude and sketch future work in section 5.

2 Fast Stixel Estimation

The stixel world model can be decomposed in three sets of parameters: the ground plane, the distance to the objects and the objects height.

For objects detection we assume the object height is known and class dependent, thus we focus on ground plane and distance to objects estimation.

Similar to previous work [14,13,2] we first estimate the ground plane (using evidence collected in v-disparity domain[1]) (see section 2.1), and then we use the ground plane to estimate the distance to the objects ("stixels distance"). The key difference of our proposal, is that the distance estimation is formulated directly in the u-v domain, instead of u-disparity as previously done (see section 2.2).

Assumptions. We share the same assumptions than [14,2]. The key assumption is that the camera has negligible roll with respect to the ground plane (pitch and height are estimated). In the current implementation we use a flat ground plane model, but nothing impedes using non-linear models. The object height is assumed known, but the method is robust to fluctuations on the height. Being stereo matching based, the common Lambertian surface assumption is used.

2.1 Ground Plane Estimation

We use the same approach as [2]. The ground plane is estimated using the v-disparity method [6], but the evidence is collected directly from matching left and right image rows at different disparities (instead of computing and projecting a depth map). See figure 2.

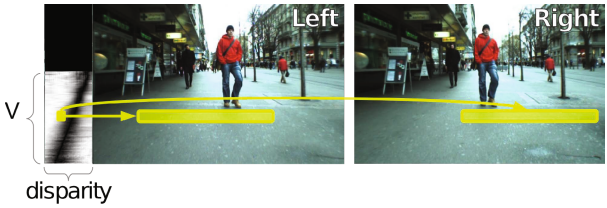


Fig. 2. The evidence for the ground plane estimation is computed by matching pixels in the left and right images

[1] In this text u refers to the image columns, and v to the image rows.

Given the evidence collected in each row, we extract the disparity with minimal cost per row and then use robust line fitting to find the ground plane. The ground plane model is represented by a function $f_{ground} : V \mapsto D$ that maps every image row v to a specific disparity $d = f_{ground}(v)$.

In order to speed-up computation, instead of collecting evidence for every row below the horizon; we consider computing only one-out-of-N row. This provides significant speed-up without degrading quality (specially when handling large images).

2.2 Stixel Distance Estimation

Previous methods for stixel distance estimation suggested to collect evidence column-wise in the image, and then estimate the distance by solving a dynamic programming problem in the u-disparity domain [14,13,2]. Although effective, this approach has three weaknesses:

Wrong quantization. When concerned about objects detection, we search to answer "where in the image can we expect objects of the class of interest?". When computing in the u-disparity domain, we create a non-homogeneous discretization in the image space (linear grows of ground plane distance, but quadratic grows of disparity distance). The quantization of the objects bottom (v dimension) will be more fine grained near the horizon, and coarser at the bottom of the image. For objects detection we rather have a regular quantization on the vertical axis.

Ignores horizontal gradient. When matching left and right images for different disparities, the information being used are the image gradients aligned to the vertical axis (vertical gradient). The horizontal gradient is completely un-informative for stereo matching. In a sense, the v-disparity cost matrix (see [2]) contains only *half* of the information available in the image.
We expect that the ground-object boundary correlates with the horizontal gradient (i.e. that more often than not, there is a visible boundary at the object bottom). This information should be exploited.

Computes more than needed. Because of the quantization effect discussed above, not only we have undesired quantizations, but also we are computing more than needed. When in the disparity domain, we will reach sub-pixel resolution close to the horizon, causing redundant computations. For objects detection, we only require to compute enough to have a coarse (and unbiased) estimate of the object position in the image.

Sampling Rows and Columns. To solve these problems we propose to *change the parametrization* from the u-disparity domain into directly the u-v domain. We are unaware of any previous work that used such parametrization.

In order to exploit the horizontal gradient information, one could weight the likelihood of a particular boundary candidate by the gradient magnitude at a particular pixel. This would force to consider every pixel below the horizon line. In order to speed-up the evaluation we use an alternative approach. The image

is split vertically in multiple row bands, and inside every band, for every image column, the pixel with the maximal horizontal gradient is selected (see figure 3). By only evaluating evidence at the pixel with maximum gradient we significantly reduce the required computation. By selecting the maximal gradient we increase the chances to find the object border accurately. In the possible but unlikely case that the ground has distracting horizontal stripes the incurred error is bounded (by the band height).

The row band i is termed b_i. The particular row selected inside band b_i at stixel q_j is termed $v\,(q_j,\, b_i)$. Given the ground plane model we can write $d\,(q_j,\, b_i) = f_{ground}\,(v\,(q_j,\, b_i))$.

Objects of interest will in general have an image width larger than one column, thus computing evidence for every column on the image is highly redundant. We allow ourselves to sample evidence each one-out-of-N column, at regular intervals. Each stixel q_j is located at column $u\,(q_j) = j \cdot stixel\_width$.

By selecting different stixel widths and row band heights, we are able to control the amount of data extracted from the image.

Dynamic Programming Formulation. The goal is to find the optimal row band for each stixel

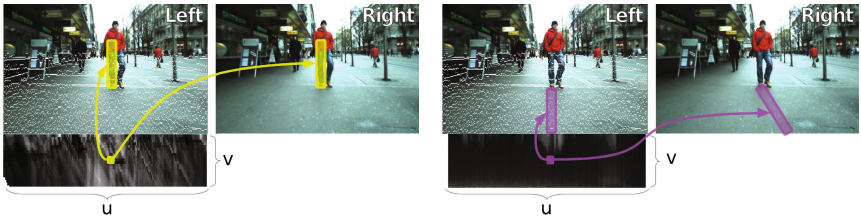
$$b_s^*(q) = \underset{b(q)}{\operatorname{argmin}} \sum_q c_s\,(q,\, b(q)) + \sum_{q_a, q_b} s_s\,(v\,(q_a,\, b(q_a))\,,\; v\,(q_b,\, b(q_b))) \qquad (1)$$

where q_a, q_b are neighbours ($|a - b| = 1$), c_s is the data term and s_s the smoothness term. This problem can efficiently be solved using dynamic programming [14,2].

Data term For each stixel column q and row band b, we calculate the evidence supporting the presence of a stixel in the left image by computing the cost $c_s(q, b)$ ("stixel cost"). The lower the cost the more likely that a stixel is present.

$$c_s(q, b) = c_o(u(q),\, d(q, b)) + c_g(u(q),\, d(q, b)) \qquad (2)$$

where $c_o(u, d)$ ("object cost"), the cost of a vertical object being present, and $c_g(u, d)$ ("ground cost"), the cost of a supporting ground being present. See figure 3 for an illustration on how these cost are computed. c_o simply sums the evidence along the vertical column, using the expected object height, projected in the image using the distance given by the ground plane estimation. c_g sums the evidence along the ground plane. In an efficient implementation, the ground plane estimate is used to warp the right image such as c_g can be computed using sums over columns between the left image and the warped right image. This warping can be done even with non linear ground plane models. See [2] for more details on c_o and c_g computation.



(a) Object evidence (b) Ground evidence

Fig. 3. The object and ground costs are computed by matching pixels in the left and right images. White dots on the image indicates object-ground boundary candidates, based on horizontal gradient maxima.

Smoothness term The smoothness term s_s enforces to respect the left-right occlusion constraints and promotes ground-object boundaries with few jumps.

$$s_s\left(v_a, v_b\right) = \begin{cases} \infty & \text{if } d\left(v_a\right) < d\left(v_b\right) - 1 \\ c_o\left(u_a, d\left(v_a\right)\right) & \text{if } d\left(v_a\right) \approx d\left(v_b\right) - 1 \\ -w \cdot c_o\left(u_a, d\left(v_a\right)\right) & \text{if } q_a = q_b \\ 0 & \text{if } d\left(v_a\right) > d\left(v_b\right) - 1 \end{cases} \tag{3}$$

The sign $\approx$ in equation 3 indicates that we will consider the lowest q_b where $s_s \neq \infty$ as if $d\left(v_a\right) = d\left(v_b\right) - 1$. w is a free parameter that promotes boundaries with few jumps, we use $w = 0.5$.

2.3 Algorithmic Speed-Up

Solving the dynamic programming problem of equation 1 has a complexity $O\left(Q \cdot B^2\right)$ where Q is the number of stixels in the image, and B is the number of row bands considered. The computation cost of the data term c_s is directly proportional to $Q \cdot B$. The total computational cost of our method is then $O\left(k_a \cdot Q \cdot B^2 + k_b \cdot Q \cdot B + k_c \cdot Q\right)$, where $k_c \cdot Q$ indicates the operations required to find the maximal gradient point for each row band. Reducing B allows to significantly improve the computation time.

In [2] every column of the image and every disparity are considered, typically, 640 columns and 128 disparities. In our implementation, for such case roughly half of the time is spent computing c_s, and half of the time solving the dynamic programming problem. If we consider stixels of width 2 pixels and 50 row bands, the total computation is expected to drop by a factor 7 (assuming $k_a = k_b$ and $k_c \ll k_b$). In section 4 we show how the quality and the effective speed fluctuate with different values for Q and B. However, it can be readily seen that significant speed-ups are achievable with this approach.

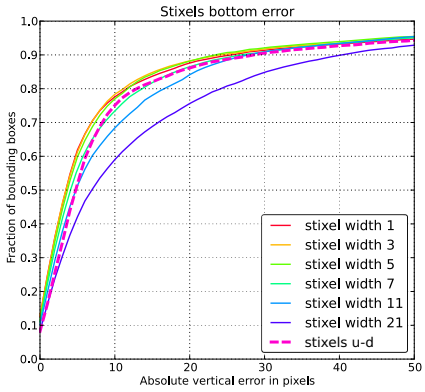
It should be noted that, as it is, we only consider objects with their bottom visible in the image. For full object detection this is a desirable property.

Nothing impedes to use the method considering row bands "out of the image" (corresponding to larger disparity values), we only need to propagate the values c_g in the visible area towards the invisible areas.

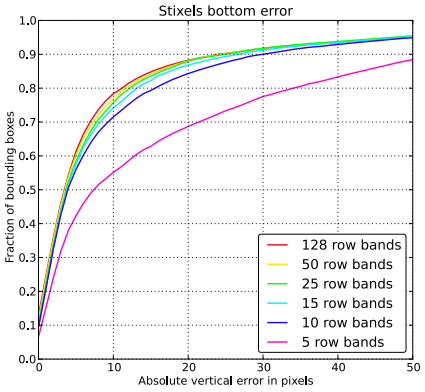
3 Object Detection Using Stixels

The authors of [1] have kindly provided an early access to their open source release of the (so called) `VeryFast` detector. When coupled with stixels estimates this detector uses the stixels of the *previous* frame, to guide the detections in the current frame. Given the expected center position of the object (stixel) in the image, and its expected scale; the detector will search around a few pixels up-and-down in the column (e.g. ± 30 pixels) and a few scales (e.g. ± 5 scales).

The number of windows evaluated is then be number of columns$\times$vertical search range$\times$scales search ranges (e.g. $640 \times 60 \times 10$), instead of every column$\times$every row$\times$every scale (e.g. $640 \times 480 \times 55$), which corresponds to a $\sim 20\times$ reduction in the search space. In section 4 we show that using the new "fast" u-v stixels estimation has no impact on quality respect to the original stixels estimation method used in [1].



(a) Varying stixel width



(b) Varying number of row bands (stixel width 3 pixels)

Fig. 4. Effect of stixel width and number of row bands on the quality

4 Evaluation

4.1 Stixels Estimation

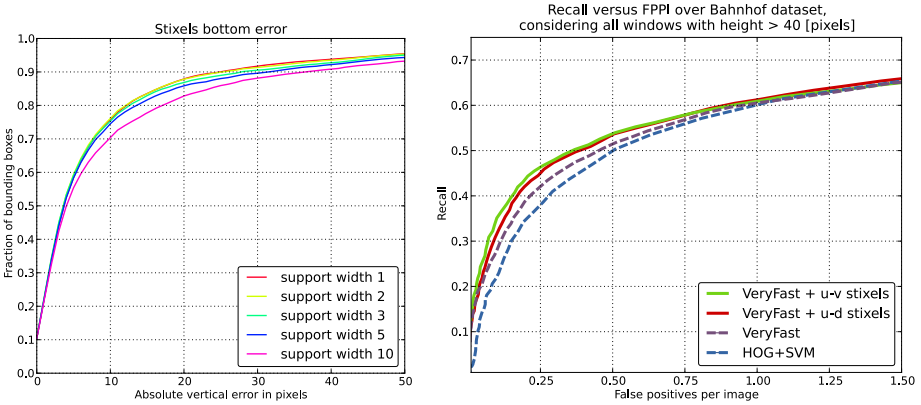
In comparison to [2] we have introduced a few additional degrees of freedom, which allow a more fine grained control of the quality versus speed trade-off. In figures 4 and 5, we use the same experimental protocol as in [2]. Given the ground

truth annotations of the pedestrians in the stereo sequence Bahnhof (999 frames, $\approx$ 7400 annotated bounding boxes), we measure the vertical distance between the estimated stixel bottom (at center column of the annotation) and the bottom of the annotated bounding box. The plots show the cumulative absolute error over the sequence (error versus fraction of bounding boxes below such error).

In figure 4a, we use a very high number of row bands (close to the number of rows below the horizon), and vary the stixel width. We see that up to stixel width of 5 pixels the quality is almost unaffected (since pedestrians are much larger than 5 pixels), but as the width further increases the quality starts to drop. It is interesting to note that computing stixels in u-v domain provides a slight quality improvement with respect to using the u-disparity domain (denominated `stixels u-d` in figure 4a). This is due to be using the horizontal gradient evidence directly which provides improved robustness to the noise in the ground plane estimate, and bypasses disparity quantization effects.

In figure 4b, we use a stixel width of 3 pixels and vary the number of row bands. Moving from 128 bands to 25 bands provides only a very small drop in quality, yet a significant improvement in speed (see section 2.3). Of course, if the number of bands is too low then quality does drop significantly.

In figure 5a we show the quality impact of using more than one column to cumulate evidence (i.e. not matching a single pixel per row, but pair of pixels, or trio of pixels, etc...). Against the intuition, we see that matching more than one column at a time does not significantly improve quality. This shows that matching only one column already collects the most relevant data from the image (and has the minimal computation cost). Matching too many columns (> 5) creates a blurring effect that degrades the quality.



(a) Varying stixel support width (stixel width 3 pixels, 25 row bands)

(b) Detection quality of different methods

Fig. 5. Stixel and detection quality results

When using a stixel width of 1 pixels and 128 row bands in our u-v stixels implementation we reach 45 Hz on a high-end laptop (4 cores of an Intel

Fig. 6. Some example frames from the result video (see supporting material). Blue lines indicate stixels estimates, boxes mark detected pedestrians.

Core i7-2630QM @ 2.00GHz). From the experiments above we see that using stixel width of 3 pixels, 25 row bands and cumulating evidence along 1 column, provides essentially the same quality as the u-disparity stixels in [2]. Using these parameters we reach 260 Hz, a 2× improvement over [2].

4.2 Object Detection

Stixels estimates can be used to speed-up objects detection. We use the open source `VeryFast` detector from [1] in the same setup that they proposed. We only did minor code optimizations to push further the execution speed by 10%. The key point of this paper is showing that we can make the detection speed GPU bounded instead of CPU bounded as in [1] (thus freeing the CPU for additional tasks).

In figure 5b, we show the detection quality obtained using the traditional `HOG+SVM` detector [17], using the `VeryFast` by itself, `VeryFast` with stixels in the u-disparity domain (`VeryFast + u-d stixels`), and the new `VeryFast` with stixels in u-v domain (`VeryFast + u-v stixels`). It can be seen that the quality is not altered, yet we move from 135 Hz [1] CPU-bound to 165 Hz GPU-bound (CPU side running at 210 Hz), average over 1000 frames on Bahnhof sequence using an Intel Core i7 870 CPU and an Nvidia GeForce GTX 470 GPU. In figure 6 we show some examples of the corresponding stixels and detection quality.

5 Conclusion and Future Work

In this paper we have proposed a new approach for obstacle detection that focuses on computing exactly what we need, and nothing more. We argue that this frugal approach is better than the previous ones because it enables estimating directly what we want ("where are the objects?"), in the domain we want ("where in the image are the objects?"), reaching the speeds we want (as fast as possible).

With our new u-v domain stixels, fast pedestrian detections on stereo image becomes (again) GPU-bound (at 165 Hz). Further speed should then be reached by improving the GPU detector itself, although our initial attempts indicate that this seems a difficult feat. In the current setup higher GPU speed seem easier to reach by hardware updates than by algorithmic modifications. We believe that exploring improvements in quality is a more fruitful direction at this time.

We are also interested in exploiting the high detection speed to build larger systems that employ pedestrian detection as an intermediate step.

Acknowledgement. This work has been partly supported by the Toyota Motor Corporation and the ERC grant COGNIMUND.

References

1. Benenson, R., Mathias, M., Timofte, R., Van Gool, L.: Pedestrian detection at 100 frames per second. In: CVPR (2012)
2. Benenson, R., Timofte, R., Van Gool, L.: Stixels estimation without depthmap computation. In: ICCV, CVVT Workshop (2011)
3. Badino, H., Franke, U., Pfeiffer, D.: The Stixel World - A Compact Medium Level Representation of the 3D-World. In: Denzler, J., Notni, G., Süße, H. (eds.) DAGM 2009. LNCS, vol. 5748, pp. 51–60. Springer, Heidelberg (2009)
4. Franke, U., Kutzbach, I.: Fast stereo based object detection for stop and go traffic. In: IVS (1996)
5. Franke, U., Joos, A.: Real-time stereo vision for urban traffic scene understanding. In: IVS (2000)
6. Labayrade, R., Aubert, D., Tarel, J.P.: Real time obstacle detection on non flat road geometry through 'v-disparity' representation. In: IVS (2002)
7. Nedevschi, S., Danescu, R., Frentiu, D., Marita, T., Oniga, F., Pocol, C., Schmidt, R., Graf, T.: High accuracy stereo vision system for far distance obstacle detection. In: IVS (2004)
8. Hu, Z., Lamosa, F., Uchimura, K.: A complete u-v-disparity study for stereovision based 3D driving environment analysis. In: 3DIM (2005)
9. Seki, A., Okutomi, M.: Robust obstacle detection in general road environment based on road extraction and pose estimation. In: IVS (2006)
10. Bajracharya, M., Moghaddam, B., Howard, A., Brennan, S., Matthies, L.H.: A fast stereo-based system for detecting and tracking pedestrians from a moving vehicle. IJRR (2009)
11. Ess, A., Leibe, B., Schindler, K., Van Gool, L.: Robust multi-person tracking from a mobile platform. PAMI (2009)
12. Okutomi, M., Noguchi, S.: Extraction of road region using stereo images. In: ICPR, vol. 1, pp. 853–856 (1998)
13. Badino, H., Franke, U., Mester, R.: Free space computation using stochastic occupancy grids and dynamic programming. In: ICCV, Workshop on Dynamical Vision (2007)
14. Kubota, S., Nakano, T., Okamoto, Y.: A global optimization algorithm for real-time on-board stereo obstacle detection systems. In: IVS, Turkey (June 2007)
15. Enzweiler, M., Hummel, M., Pfeiffer, D., Franke, U.: Efficient stixel-based object recognition. In: IVS (2012)
16. Günyel, B., Benenson, R., Timofte, R., Van Gool, L.: Stixels Motion Estimation without Optical Flow Computation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part VI. LNCS, vol. 7577, pp. 528–539. Springer, Heidelberg (2012)
17. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)