Instance stixels

Segmenting and grouping stixels into objects

Hehn, Thomas; Kooij, Julian; Gavrila, Dariu

DOI
10.1109/IVS.2019.8814243

Publication date
2019

Document Version
Final published version

Published in
Proceedings IEEE Symposium Intelligent Vehicles (IV 2019, Paris)

2019 IEEE Intelligent Vehicles Symposium (IV)
Paris, France. June 9-12, 2019

Instance Stixels: Segmenting and Grouping Stixels into Objects

Thomas M. Hehn, Julian F. P. Kooij and Dariu M. Gavrila[1]

***Abstract*— State-of-the-art stixel methods fuse dense stereo and semantic class information, e.g. from a Convolutional Neural Network (CNN), into a compact representation of driveable space, obstacles, and background. However, they do not explicitly differentiate instances within the same class. We investigate several ways to augment single-frame stixels with instance information, which can similarly be extracted by a CNN from the color input. As a result, our novel Instance Stixels method efficiently computes stixels that do account for boundaries of individual objects, and represents individual instances as grouped stixels that express connectivity. Experiments on Cityscapes demonstrate that including instance information into the stixel computation itself, rather than as a post-processing step, increases Instance AP performance with approximately the same number of stixels. Qualitative results confirm that segmentation improves, especially for overlapping objects of the same class. Additional tests with ground truth instead of CNN output show that the approach has potential for even larger gains. Our Instance Stixels software is made freely available for non-commercial research purposes.**

I. Introduction

Self-driving vehicles require a detailed understanding of their environment in order to react and avoid obstacles as well as to find their path towards their final destination. In particular, stereo vision sensors obtain pixel-wise 3D location information about the surrounding, providing valuable spatial information on nearby free space and obstacles. However, as processing power is a valuable resource, it is essential to find a compact representation of sensor measurements which is still capable to provide adequate information about the environment [2], [3]. A common approach is to create a dynamic occupancy grid for sensor fusion [4] and tracking [5], which provides a top-down grid cell representation of occupied space surrounding the ego-vehicle. Still, directly aggregating depth values into an occupancy grid alone would disregard the rich semantic information from the intensity image, and the ability to exploit the local neighborhood to filter noise in the depth image.

A popular alternative in the intelligent vehicles domain is the "stixel" representation, which exploits the image structure to reduce disparity artifacts, and is computed efficiently [6]. By grouping pixels into rectangular, column-wise superpixels based on the disparity information, stixels reduce the complexity of the stereo information. Recently, class label information obtained from deep learning has been incorporated into the stixel computation and representation, so-called Semantic Stixels [1]. Yet, they are still a loose collection of upright "sticks" on an estimated ground plane, lacking object level information.

[1] All authors are with the Intelligent Vehicles Group, TU Delft, The Netherlands. Primary contact: t.m.hehn@tudelft.nl

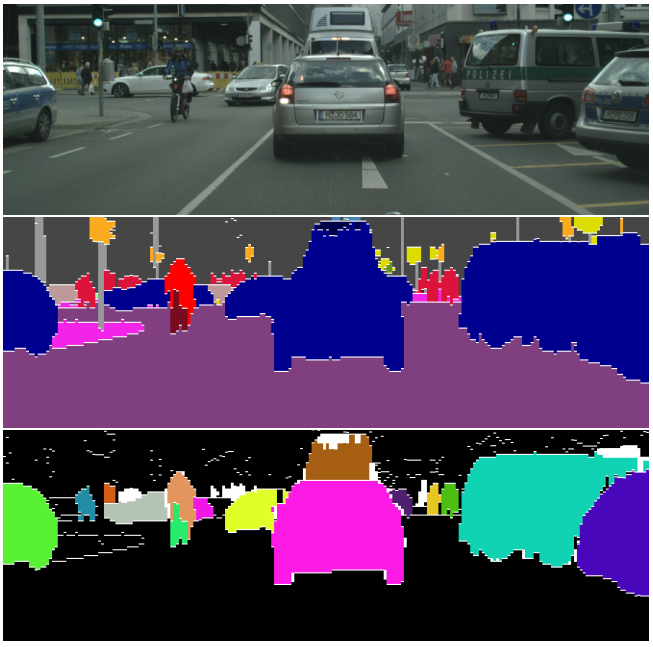


Fig. 1: Top: Input RGB image (corresponding disparity image not shown). Middle: Semantic Stixels [1] use a semantic segmentation CNN to create a compact stixel representation which accounts for class boundaries (horizontal stixel borders: white lines, vertical borders: not shown; arbitrary colors per class). Note that a single stixel sometimes covers multiple instances, e.g. multiple cars. Bottom: Our novel *Instance Stixels* algorithm also accounts for instance boundaries using additional information learned by a CNN and clusters stixels into coherent objects (arbitrary colors per instance).

This paper introduces an object level environment representation extracted from stereo vision data based on stixels. Our method improves upon state-of-the-art stixel methods [1], [7] that only consider semantic class information, by adding instance information extracted with a convolutional neural networks (CNN) from the input RGB image. This provides several benefits. First, by fusing disparity, semantic, and instance information in the stixel computation, we obtain better stixels boundaries around objects. Second, the instance information in each stixel is then used in a post-processing step to efficiently group stixels into distinct objects, which facilitates subsequent scene analysis steps.

II. Related work

The idea of stixels, regarding objects as sticks perpendicular on a ground plane was introduced by [6]. The stixel algorithm has found diverse application in the autonomous

978-1-7281-0560-4/19/$31.00 ©2019 IEEE

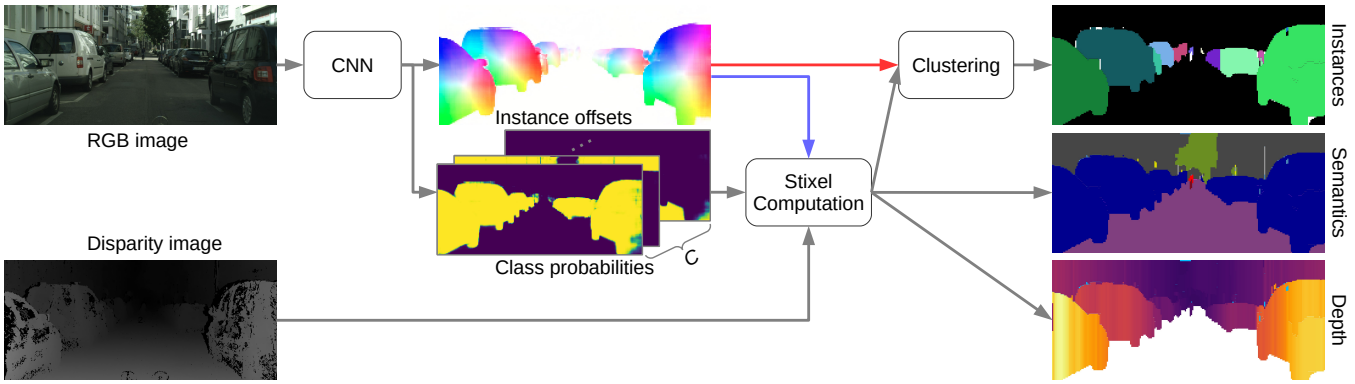


Fig. 2: Instance stixel pipeline applied to a *RGB* and *disparity* input image pair obtained from a stereo camera. The RGB image is processed by a Convolutional Neural Network (*CNN*) to predict *offsets* to the instance centers (HSV color coded) and per-pixel semantic *class probabilities* (visualized as color gradient). The class probabilities are fused with the disparity input image in the *Stixel computation* to provide a super-pixel representation of the traffic scene, which unifies *Semantics*, *Depth* and additionally *Instance* output (left images). In the baseline algorithm (*Semantic Stixels + Instance*, red arrow) the obtained stixels are clustered based on the instance offsets to assign stixels to instances (not shown). In contrast, our proposed algorithm (*Instance Stixels*, blue arrow) fuses the instance offset information with the other two channels in the Stixel Computation. Subsequently, stixels are also clustered to form instances, but with improved adherence of stixels to instance boundaries (top right image, arbitrary colors).

driving domain. [8] used stixels to detect of small unknown objects, such as lost cargo. In [9] the idea was further extended to a multi-layer representation using a probabilistic approach, i.e. stixels do not need to be connected to the ground plane anymore. In the multi-layer representation, stixels segment the entire image into rectangular super-pixels, classified as ground, object or sky. Additionally, a dynamic programming scheme was presented for efficient real-time computation of stixels. For even faster computation, this dynamic programming scheme was then also implemented on the Graphical Processing Unit (GPU) by [10]. In [11] stixels were compared with other super-pixel algorithms as basis for multi-cue scene labeling.

The general stixel framework offers various possibilities for extension. Driven by the requirements of autonomous driving, [12] introduced applied a Kalman filter to track single stixels. Stixel tracking was then further improved by [13]. Yet, stixels are generally tracked independently and not as parts of an object. In order to obtain object information [14], [15], [16], [17] group the Dynamic Stixels based on shape cues and graph cuts and thus rely on tracking Stixels over time. Stixels are also applied in semantic scene segmentation with more general classes than ground, object and sky. For this purpose, semantic information can be obtained by using object detectors for suitable classes [18] or Random Decision Forest classifiers [19] and then including that information in the Stixel generation process. [1] extend this idea by incorporating the output of a Fully Convolutional Neural Network (FCN) in the probabilistic stixel framework. They named their approach Semantic Stixels. Based on Semantic Stixels and focusing on non-flat road scenarios, [7] generalize stixels to also model slanted surfaces, e.g. not strictly perpendicular to the road anymore, including piece-wise linear road surfaces.

Meanwhile, many more deep neural network architectures have been proposed in the computer vision literature to improve classification and image segmentation tasks. For instance, Residual Neural Networks [20] facilitate training of deeper networks by learning residual functions. Dilated Residual Networks [21] (DRN) improve on this work by maintaining a higher resolution throughout the fully connected network, while working with the same receptive field. As a consequence, they are useful for applications that require spatial reasoning such as object detection or, as in our case, instance segmentation. Unfortunately, one cannot treat instance segmentation as a classification problem, as is done for semantic segmentation. A main reason is that the number of instances varies per image, which prohibits a one-to-one mapping of network output channels to instances. Instead of predicting instance labels directly, [22] trains a CNN to map each pixels in an image onto a learned low-dimensional space such that pixels from the same instance map close together. Object masks are then obtained in post-processing by assigning pixels to cluster centers in this space. The Box2Pix method [23] instead uses supervised learning to map pixels to a specific target space, namely the 2D offsets from the given pixel towards its instance's center, which are found through a bounding box detection branch.

Our objective is to create efficient stixel representations rather than pixel-accurate instance segmentation in images, and to avoid overhead of clustering all pixels into instances before reducing them to a compact representation. Still, we follow insights from the work on per-pixel instance segmentation to improve stixel computation, deal with the unknown number of instances in an image, and enable the clustering of stixels into instances. The main contributions

are thus summarized as:

- We present Instance Stixels, a method to include instance information into stixels computation, which creates better stixels, and allows grouping to instance IDs from a single stereo frame.
- We investigate two different ways to include the instance information, and show that adding the information into the stixel computation itself results in more accurate instance representations than only using it to cluster Semantic Stixels.
- Our software for Semantic Stixels and Instance Stixels is provided as open-source to the scientific community for non-commercial research purposes.

III. Methods

This section will first shortly introduce the original disparity Stixel and Semantic Stixel formulations in subsection III-A. Subsection III-B then explains how to integrate the instance information from a trained CNN into the stixel computation itself for improved stixel segmentation. Finally, subsection III-C will discuss how the instance information can be used to cluster stixels belonging to the same object.

The clustering step could be applied to any stixel computation method. We therefore consider two options:

- Clustering stixels from a standard Semantic Stixels method [1], such that instance offset information is only considered here at this final clustering step. This corresponds to the red arrow in Figure 2. In our experiments we shall refer to this combination as the **Semantic Stixels + Instance** method.
- Clustering based on our novel instance-aware stixels computation from section III-B, see the blue arrow in Figure 2. We name this combination our novel **Instance Stixels** method.

Conceptually, Instance Stixels are a natural extension to Semantic Stixels as they extend disparity and semantic segmentation information with additional instance information to compute a compact representation from a stereo image pair. These stixels also receive an object id which groups them into instances.

A. Stixels

In the following, an outline of the derivation of the original Stixels and Semantic Stixels framework is presented. For a more detailed derivation, see [24] and [1].

1) Disparity Stixels: Following the notation of [24], the full stixel segmentation of an image is denoted as $L = \{L_u | 0 \leq u < w\}$ with w being the total number of stixel columns in the image. The segmentation of column u contains $L_u = \{s_n | 1 \leq n \leq N_u \leq h\}$ contains at least one but at most height h stixels s_n. A stixel $s_n = (v_n^b, v_n^t, c_n, f_n(v))$ is described by the bottom and top rows, respectively v_n^b and v_n^t, that delimit the stixel. Additionally, a stixel is associated with a class $c_n \in \{g, o, s\}$ (i.e. ground, object, sky) and a function f_n which maps each row of the image to an estimated disparity value.

The aim is to find the stixel segmentation L which maximizes the posterior probability of L given an input measurement (e.g. a disparity image) D. Since each column $u \in \{0, ..., w-1\}$ of the image is treated independently, the MAP objective can then be written as:

$$\arg\max_{L} \prod_{u=0}^{w-1} p(D_u | L_u) p(L_u). \tag{1}$$

Here, $p(D_u|L_u)$ denotes the column's likelihood of the disparity data, and $p(L_u)$ is a prior term [9].

Assuming all rows are equally likely to separate two stixels, the column likelihood term can be written as product of individual terms for N_u stixels, $L_u = \{s_1, ..., s_{N_u}\}$. Since only disparity values of the rows within each stixel contribute to its likelihood, those terms can in turn be factorized over the rows $v_n^b \leq v \leq v_n^t$ of each stixel $n \in \{1, ..., N_u\}$. Hence, the final objective is [24]:

$$\arg\max_{L} \prod_{u=0}^{w-1} \prod_{n=1}^{N_u} \prod_{v=v_n^b}^{v_n^t} p(d_v | s_n, v) p(L_u). \tag{2}$$

In practice, this objective is written as an energy *minimization* problem by turning the product over probabilities into a sum of negative log probabilities, which is then solved efficiently through Dynamic Programming (DP) [24], [10]. DP will evaluate the energy function for many stixel hypotheses $L_u = \{s_1, ..., s_{N_u}\}$, which consists of unary terms $E_d(s_n)$ and pairwise energy terms $E_p(s_{n-1}, s_n)$. Intuitively, the unary energy term $E_d(s_n)$ describes the disparity 'error' within a stixel.

2) Semantic Stixels: The Semantic Stixels method [1] introduced an additional semantic data term to associate each stixel with one class label $l_n \in \{1, ..., C\}$. Thus, Semantic Stixels are characterized by $s_n = (v_n^b, v_n^t, c_n, f_n(v), l_n)$. First, a semantic segmentation CNN is trained on RGB images with annotated per-pixel class labels. Then, when testing on a test image, the softmax outputs $\sigma(p, l)$ for all semantic classes l of all pixels p are kept (note that in a standard semantic segmentation task, only the class label of the strongest softmax output would be kept). The unary data term $E_d(s_n)$ of the original disparity stixel computation is then replaced by $E_u(s_n) = E_d(s_n) + \omega_l E_l(s_n)$, thereby adding semantic information from the network activations,

$$E_l(s_n) = -\sum_{p \in \mathcal{P}_n} \log \sigma(p, l_n). \tag{3}$$

Here $\mathcal{P}_n$ are all pixels in stixel s_n, and ω_l a weight factor.

B. Instance Stixels

Instance Stixels expand the idea of Semantic Stixels by additionally training a CNN to output a 2D estimation of the position of the instance center for each pixel. This estimation is predicted in image coordinates, as proposed in the Box2Pix method [23]. More specifically, the CNN predicts 2D offsets $\mathbf{\Omega}_p \in \mathbb{R}^2$ (i.e. x and y direction) per pixel, which are relative to the pixel's location in the image. As a consequence, for all pixels p belonging to the same instance j, adding their

ground truth offset $\hat{\boldsymbol{\Omega}}_p$ to the pixel location (x_p, y_p) will end up at the same instance center $\hat{\boldsymbol{\mu}}_j$:

$$\hat{\boldsymbol{\mu}}_j = \hat{\boldsymbol{\Omega}}_p + (x_p, y_p). \tag{4}$$

We refer to such as a network as the *Offset CNN*, and an example of its output and ground truth training data are visualized in figures 3c and 3d. The ground truth instance centers are defined as the center of mass of the ground truth instance masks. Note that instances are commonly only considered for certain semantic classes, e.g. cars, pedestrians and bicycles. For all other classes, the target offset is $(0, 0)$.

Instance Stixels then incorporate the Offset CNN output into the stixel computation. Let $\boldsymbol{\mu}_p$ denote the instance center estimate obtained from the CNN for some pixel p, and $\bar{\boldsymbol{\mu}}_n = \sum_{p \in \mathcal{P}_n} \boldsymbol{\mu}_p$ the mean over all pixels in an instance stixel $s_n = (v_n^b, v_n^t, c_n, f_n(v), l_n, \bar{\boldsymbol{\mu}}_n)$. Furthermore, let $\mathcal{I} \subset \mathbb{N}$ denote the set of classes for which stixels need to be assigned to instances. We model the instance term depending on the center estimates of the pixels and the mean instance center of the current stixel hypothesis s_n:

$$E_i(s_n) = \begin{cases} \sum_{p \in \mathcal{P}_n} ||\boldsymbol{\mu}_p - \bar{\boldsymbol{\mu}}_n||_2^2, & \text{if } l_n \in \mathcal{I} \\ \sum_{p \in \mathcal{P}_n} ||\boldsymbol{\mu}_p - (x_p, y_p)||_2^2, & \text{otherwise.} \end{cases} \tag{5}$$

In other words, for instance classes, the instance term favors stixels which cover an area that consistently points to the same instance center. For non-instance classes, i.e. classes not in $\mathcal{I}$, offsets deviating from zero contribute to the instance energy. Without this, classes with instance information would generally have higher energy and thus be less likely than the non-instance classes.

With the instance energy term, the unary energy becomes

$$E_u(s_n) = \omega_d E_d(s_n) + \omega_s E_s(s_n) + \omega_i E_i(s_n). \tag{6}$$

This also introduces weights ω_d and ω_i for the disparity and instance terms for more control on the segmentation.

A useful side effect is that each instance stixel already has a mean estimate of its instance center pixel coordinates, which will be used when clustering stixels into objects, which will be discussed in Section III-C.

C. Clustering stixels with instance information

We now describe how output from an Offset CNN can be used in a post-processing step to cluster stixels. Note the favorable computational complexity of grouping a low number of stixels rather than individual pixels as in conventional instance segmentation tasks.

First, the per-pixel offsets from the Offset CNN are aggregated into a per-stixel offset estimate by averaging the CNN's predictions over the pixels in the stixel (this is already done for Instance Stixels, as noted in Section III-B). Hence, each stixel is equipped with an estimate of its instance center in 2D image coordinates, as well as a semantic class label.

Then, the estimated instance centers and semantic class prediction are used to group stixels to form instances. Separately for each semantic class, we aim to find clusters in the estimated instance centers. For this purpose, we resort to the DBSCAN clustering algorithm [25] as it estimates the number of clusters (i.e. instances) and performs well when the data has dense clusters. DBSCAN has only two parameters: the maximum distance between neighboring data points ε and the minimum size γ of the neighborhood of a data point in order to consider this point a core point. Additionally, we introduce a size filter parameter which prevents stixels that are smaller (i.e. cover less rows) than ρ to be considered a core point.

IV. Implementation

To develop and test our ideas, we extended the open-source disparity Stixel CUDA-implementation introduced in [10]. Our extensions implement both the computation of Semantic Stixels according to [1], as well as the Instance Stixel method presented here. Further, handling of invalid disparity measurements was implemented. This implementation reduces the input image before the stixel optimization to a column image. I.e. the stixel optimization is based only on the mean value of a row of the column. As a consequence, a row in a column is only considered as invalid disparity measurement, if all pixels in that row are invalid.

Finally, we note that any semantic segmentation network architecture could be used for Semantic Segmentation and Offset CNN. We use Dilated Residual Networks [21] (DRN) as our underlying architecture in our implementation, as they have favorable properties for these tasks, as discussed in Section II. The implementation of the DRN is largely based on the PyTorch [26] code provided by the authors of [21]. The source code is available online [1].

V. Experiments

A. Dataset, metrics and pre-processing

The computation of stixels requires RGB camera images and the corresponding disparity images obtained from a stereo camera setup. We use the Cityscapes dataset [27] for our experiments, as it consists of challenging traffic scenarios. Further, it provides ground truth annotations for semantic and instance segmentation. The performance on these two tasks is evaluated using the standard Cityscapes metrics described in the corresponding paper [27].

Semantic segmentation performance is measured by the intersection-over-union (IoU) = $\frac{TP}{TP+FP+FN}$, where TP, FP, and FN denote the number of true positives, false positives and false negatives over all pixels in the dataset split. An instance mask is considered correct as the overlap with its ground truth mask surpasses a specific threshold. The Average Precision (AP) corresponds to an average over the precision for multiple thresholds. Average Precision ($AP^{50\%}$) only considers an overlap of at least 50% as true positive. The metric also allows to provide confidence score for each instance mask. We did not make use of this option and always set the confidence score to 1 for all compared algorithms.

The disparity images provided in the Cityscapes dataset exhibit noisy regions introduced due to bad disparity measurements at the vertical image edges and the hood of the car.

[1] https://github.com/tudelft-iv/instance-stixels

Inaccurate disparity data may harm the performance of disparity based Stixels. Although Semantic Stixels are already more robust due to the second modality, we aim to suppress such effects. Therefore, we crop all images symmetrically (top: 120px, bottom: 120px, left: 128px, right: 128px) to ensure that our experiments are not influenced by disparity errors. Following [1], we are using the official validation as test set. Therefore, we split the official training set into a separate training *subtrain* and validation set *subtrainval* (validation cities: Hanover, Krefeld, Stuttgart).

B. Training the Offset CNN

As shown in [23], semantic segmentation networks can be trained to simultaneously predict class-wise probabilities and instance offsets (described in section III-C). To focus our experiments on the influence of the offset prediction, we use a separate Dilated Residual Network [21] with the same architecture (*drn_d_22*), which was already trained on Cityscapes for semantic segmentation. It achieves competitive performance on the Cityscapes validation set (66.8% IoU).

The offset CNN is based on the aforementioned pretrained semantic segmentation network. We only replace the last convolutional layer which outputs a channel for each class by a convolutional layer that outputs our two offset channels. The network is then trained with a common Mean Squared Error loss function

$$\frac{1}{|\mathcal{P}|}\sum_{p\in\mathcal{P}}||\mathbf{\Omega}_p-\hat{\mathbf{\Omega}}_p||_2^2, \tag{7}$$

where $\mathcal{P}$ denotes the set of all pixels in a single image, $\mathbf{\Omega}_p$ the CNN 2D output and $\hat{\mathbf{\Omega}}_p$ the ground truth offset. We minimize the loss function using the Adam optimization [28].

It is important to note that the output of the CNN is downscaled by a factor 8. We also downscale the ground truth by that factor for training. The reason for this is that upscaling, unless nearest neighbor upscaling is used, introduces interpolation errors that result in a smooth transition of the offset vectors between two instances. As a consequence, this would also result in an interpolation of the predicted means of two neighboring instances at pixels close to the borders, which in the end yields worse clustering results. To overcome this issue, we use nearest neighbor upscaling when passing the predicted images to the Stixel algorithms. The loss of resolution is compensated by the fact that our Stixels work at a resolution of width 8.

C. Hyperparameter optimization

The algorithms we evaluate offer several hyperparameters that require tuning: the weighting of the data terms for the Stixel computation ω_d, ω_s and ω_i, as well as the DBSCAN parameters ϵ, γ and ρ. The stixel framework provides more parameters from which we set the stixel width to 8 pixels throughout our experiments. Remaining parameters are set based on recommendations from [24] and [10].

The parameter tuning is performed using Bayesian optimization [29], [30] on the *subtrainval* validation set for 100 iterations. The cost is computed as IoU$+1.5\cdot$Instance AP. We weighted Instance AP higher as this is our main focus. The optimization is performed separately for each experiment that requires clustering. The experiments that assign instances based on ground truth masks are performed using the same stixel weights that were found using clustering.

Algorithm	SS+I	IS	SS+I	IS	IS	IS
GT offsets	-	-	-	-	✓	✓
GT assignment	-	-	✓	✓	-	✓
Semantic IoU [%]	**65.2**	**65.2**	65.2	65.2	67.6	67.6
Instance AP [%]	10.4	**11.4**	18.6	20.2	24.9	34.6
Instance $AP^{50\%}$ [%]	20.2	**21.8**	41.7	43.1	39.6	57.2
Avg. # of Stixels	1346	1375	1346	1375	2331	2331

TABLE I: Quantitative evaluation of *Semantic Stixels + Instance* (SS+I, baseline) and *Instance Stixels* (IS, our proposed algorithm) on the Cityscapes validation set. Both algorithms are evaluated regarding semantic intersection-over-union (IoU), instance average precision (AP) and average number of stixels (best results in boldface, see section V-A). To further assess the potential of the algorithms, ground truth information is incorporated at different levels (i.e. *offset* and *assignment*, see section V-D). Note that only the results of *Semantic Stixels + Instance* with *ground truth assignment* are shown as they represent the upper limit of the baseline for the different ground truth variations.

D. Comparison of algorithmic variations

In order to analyze the capabilities of our proposed method, we vary three different aspects of the proposed methods that can be combined arbitrarily:

1) Semantic Stixels + Instance v. Instance Stixels: Corresponds to setting $\omega_i = 0$, which resembles Semantic Stixels [1]), or $\omega_i > 0$ in the stixels computation (see equation 6), respectively.
2) Clustering v. GT assignment: Each stixel is assigned to an instance either by applying DBSCAN on the mean estimated center per stixel or using the ground truth (GT) instance masks. For GT assignment, a stixel is assigned to the corresponding instance if the ground truth mask covers more than 10% of the stixel.
3) Predicted offsets v. GT offsets: the offsets used for generating stixels and for clustering are either predicted by the CNN or computed from the ground truth (GT) instance masks. Note: this does not affect the semantic predictions of the CNN which always remain the same.

1) Quantitative evaluation: Table I lists the results of different variations (as described in section V-D) of stixel algorithms, with respect to possible use of GT (cf. first two table rows). The first and second columns relate to our base line and the proposed method; they show that adding the instance term (equation 6) does not harm semantic performance, but increases instance AP performance by 9.6%. Note that the average number of stixels does not increase significantly, which would indicate over-segmentation. Assigning the instance IDs based on the ground truth instance masks (third and fourth column) gives insight on the clustering

performance. When compared to the algorithms without any ground truth added, one can observe a significant improvement. The last two columns should be compared to the second and forth column to understand which performance may be achieved by improving the predictions of the Offset CNN. When replacing the Offset CNN predictions by ground truth, the Instance performance improves for both clustering and ground truth assignment. Notably, using ground truth offsets also improves the semantic performance which is reasonable as offsets provide another clue to distinguish instance and non-instance classes. The increase in the average number of stixels, however, indicates over-segmentation. Including the average number of stixels in the parameters optimization may reduce this effect.

The key insights of the above results are that adding the instance term to the stixel computation boosts instance precision without weakening semantic performance. Further, we see that both, the Offset CNN and clustering, may improve the overall result. Interestingly, although given ground truth offset input, there is still a considerable gap to assigning stixels based on ground truth instance masks.

2) Qualitative analysis: The consequences of the different algorithm variations as described in section V-D are depicted in figure 3 when applied on validation images of real traffic situations. Figures 3a and 3b show the input data. Based on the RGB image 3a, the Offset CNN predicts offsets as illustrated in figure 3c. Upon inspection of the corresponding ground truth offsets shown in figure 3d some incorrect predictions may be still be discovered. In figures 3e-3j we contrast the influence of the different algorithms. Specifically, figures 3e, 3g and 3i show the algorithms that rely on ground truth masks for instance assignment. Figures 3f, 3h and 3j depict the respective result based on clustering assignment. Here, especially for small instances far away, it can be observed that the clustering fails distinguishing them. Comparing 3f and 3h demonstrates the effectiveness of Instance Stixels for partly occluded instances.

Figure 4 again contrasts the results of Semantic Stixels + Instance information compared to Instance Stixels in a real traffic scene. In particular, inspection of the top down views, figures 4e and 4f, shows that the improved coherence with the instance boundaries also reduces the noise in depth estimation. The top-down view also shows how Instance Stixels express objects as polygons, rather than unconnected points as outputted by other single-frame stixel methods [9], [1], [7]. This could benefit subsequent uses of the representation, such object tracking or fusion in a rasterized occupancy grid.

VI. Conclusions

This paper introduced Instance Stixels, which improve stixel segmentation by considering instance information from a CNN, and performing a subsequent stixel clustering step using the averaged instance information within each stixel. Our experiments showed the benefit of including the instance information already in the segmentation step, as opposed to only cluster stixels from Semantic Stixels. Quantitative results support the qualitative analysis as they show that Instance Stixels adhere better to object boundaries. This leads to almost 10% increase in Instance AP (from 10.4% to 11.4%) compared to clustering only, without significantly increasing the number of stixels. Further experiments using ground truth instance information in both the stixel computation, and the clustering, showed that the approach has the potential for even larger gains.

Acknowledgment

This work received support from the Dutch Science Foundation NWO-TTW within the Sensing, Mapping and Localization project (nr. 14892). We further thank the authors of [10] and [21] for kindly providing their code and pre-trained CNN models to the scientific community.

References

[1] L. Schneider, M. Cordts, T. Rehfeld, D. Pfeiffer, M. Enzweiler, U. Franke, M. Pollefeys, and S. Roth, "Semantic Stixels: Depth is not enough," in *Proc. of IEEE Intelligent Vehicles Symposium*, vol. 2016-Augus, 2016, pp. 110–117.

[2] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Trans. on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[3] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "Eurocity persons: A novel benchmark for person detection in automotive context," *Accepted for IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2019. [Online]. Available: http://arxiv.org/abs/1805.07193

[4] D. Nuss, T. Yuan, G. Krehl, M. Stübler, S. Reuter, and K. Dietmayer, "Fusion of laser and radar sensor data with a sequential monte carlo bayesian occupancy filter," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1074–1081.

[5] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Trans. on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1331–1342, 2011.

[6] H. Badino, U. Franke, and D. Pfeiffer, "The stixel world - a compact medium level representation of the 3d-world," in *Proc. of the 31st DAGM Symposium on Pattern Recognition*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 51–60.

[7] D. Hernandez-Juarez, L. Schneider, A. Espinosa, D. Vázquez, A. M. López, U. Franke, M. Pollefeys, and J. C. Moure, "Slanted stixels: Representing san franciscos steepest streets," *Procedings of the British Machine Vision Conference 2011*, 2018.

[8] S. Ramos, S. Gehrig, P. Pinggera, U. Franke, and C. Rother, "Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling," *Proc. of IEEE Intelligent Vehicles Symposium*, no. Iv, pp. 1025–1032, 2017.

[9] D. Pfeiffer and U. Franke, "Towards a Global Optimal Multi-Layer Stixel Representation of Dense 3D Data," *Procedings of the British Machine Vision Conference 2011*, pp. 51.1–51.12, 2011.

[10] D. Hernandez-Juarez, A. Espinosa, J. C. Moure, D. Vázquez, and A. M. López, "GPU-Accelerated real-Time stixel computation," *Proc. of IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1054–1062, 2017.

[11] M. Cordts, T. Rehfeld, M. Enzweiler, U. Franke, and S. Roth, "Tree-structured models for efficient multi-cue scene labeling," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1444–1454, 2017.

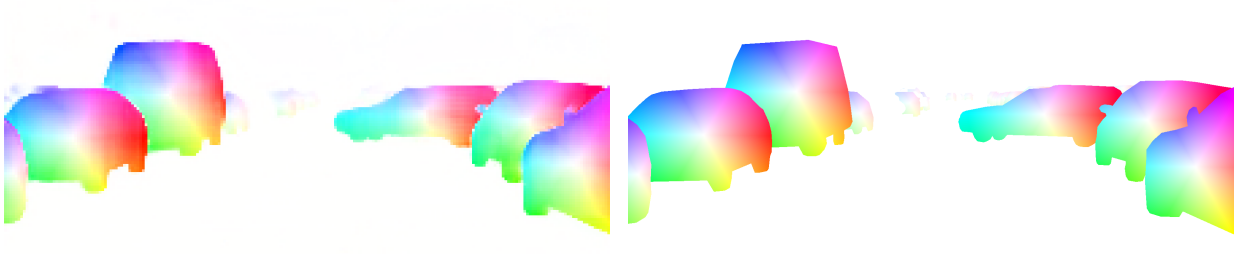
[12] D. Pfeiffer and U. Franke, "Efficient representation of traffic scenes by means of dynamic stixels," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2010, pp. 217–224.

[13] B. Günyel, R. Benenson, R. Timofte, and L. Van Gool, "Stixels Motion Estimation without Optical Flow Computation," in *Proc. of the European Conference on Computer Vision (ECCV)*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 528–539.

[14] F. Erbs, A. Barth, and U. Franke, "Moving vehicle detection by optimal segmentation of the dynamic stixel world," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 951–956.

(a) Input: Original image.

(b) Input: Disparity image.

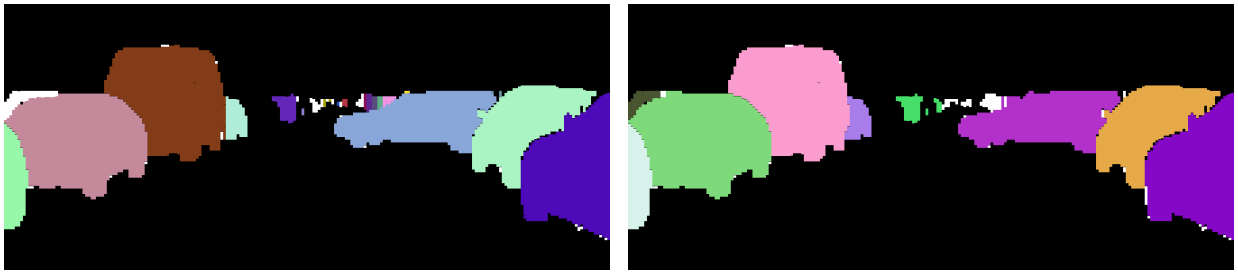


(c) CNN output: Predicted offsets.

(d) Ground Truth (GT) offsets.

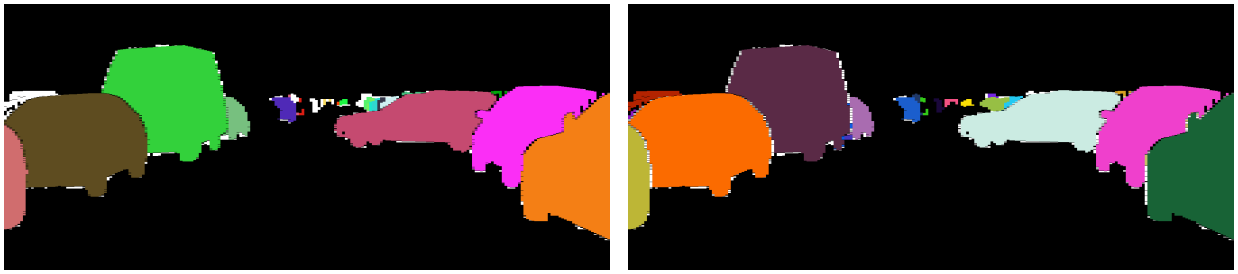
(e) Semantic stixels + Instances: Predicted offsets, GT assignment.

(f) Semantic stixels + Instances: Predicted offsets, clustered.



(g) Instance stixels: Predicted offsets, GT assignment.

(h) Instance stixels: Predicted offsets, clustered. *Proposed method, no GT.*



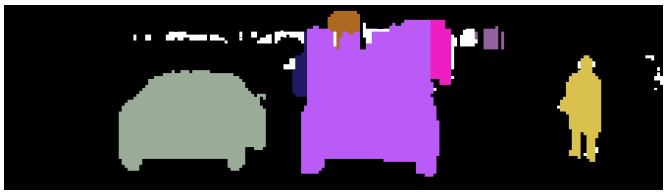
(i) Instance stixels: GT offsets, GT assignment.

(j) Instance stixels: GT offsets, clustered.

Fig. 3: Qualitative analysis of *Semantic Stixels + Instance* (baseline) and *Instance Stixels* (proposed algorithm). Figure (a) and (b): The input RGB and disparity image. Figure (c) and (d): The output of the Offset CNN and the corresponding ground truth. The direction and magnitude of the offset are indicated using HSV colorspace. Figures (e) to (j): Illustration of the instance segmentation of the baseline and our proposed method including different ground truth information (see section V-D, table I). Instances are indicated by arbitrary colors. White areas denote stixels that cannot be assigned to specific instances, but may be part of an instance due to their predicted semantic class. Left/right: GT assignment v. clustering. (e) and (f): baseline. (g) and (h): our proposed algorithm. (i) and (j): our proposed algorithm using GT offsets.

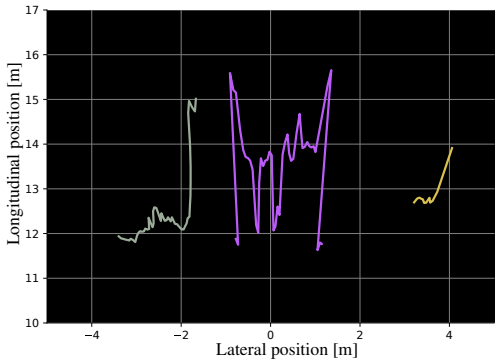
(a) Input RGB image.

(b) Input disparity image.

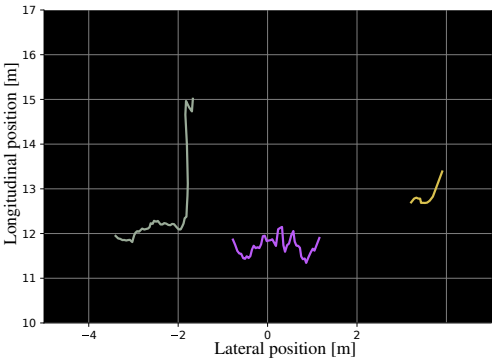


(c) Semantic Stixels + Instance.

(d) Instance Stixels.



(e) Semantic Stixels + Instance: Top down view.



(f) Instance Stixels: Top down view.

Fig. 4: Qualitative comparison of *Semantic Stixels + Instance* (baseline) and *Instance Stixels* (our proposed algorithm) with respect to depth estimation. (a) and (b): Input RGB and disparity image. (c) and (d): Instance segmentations resulting from the two algorithms. In our proposed algorithm, the stixels match the shape of the car in the image center better than in the baseline segmentation. (e) and (f): Top down views based on disparity estimation from stixels. Given the disparity, the 3D position of each stixel is computed and only the lateral and longitudinal position from the camera is plotted. Only stixels which are considered core points of an instance in the DBSCAN clustering are shown and connected. No ground truth was used. The longitudinal position estimation of our proposed algorithm is more stable than of the baseline.

[15] F. Erbs, B. Schwarz, and U. Franke, "Stixmentation-probabilistic stixel based traffic scene labeling." in *BMVC*, 2012, pp. 1–12.

[16] ——, "From stixels to objects - a conditional random field based approach," in *IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 586–591.

[17] F. Erbs, A. Witte, T. Scharwächter, R. Mester, and U. Franke, "Spider-based stixel object segmentation," in *IEEE Intelligent Vehicles Symposium (IV)*, 2014, pp. 906–911.

[18] M. Cordts, L. Schneider, M. Enzweiler, U. Franke, and S. Roth, "Object-level priors for stixel generation," in *German Conference on Pattern Recognition*. Springer, 2014, pp. 172–183.

[19] T. Scharwächter and U. Franke, "Low-level fusion of color, texture and depth for robust road scene understanding," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 599–604.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[21] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," in *Deep Learning for Robotic Vision, workshop at CVPR 2017*. CVPR, 2017, pp. 1–2.

[23] J. Uhrig, E. Rehder, B. Fröhlich, U. Franke, and T. Brox, "Box2pix: Single-shot instance segmentation by assigning pixels to object boxes," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[24] D. Pfeiffer, "The stixel world," Ph.D. dissertation, Humboldt-Universität zu Berlin, 2012.

[25] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[27] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2014.

[29] J. Mockus, V. Tiesis, and A. Zilinskas, *The application of Bayesian methods for seeking the extremum*, 09 1978, vol. 2, pp. 117–129.

[30] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.