2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Box2Pix: Single-Shot Instance Segmentation by Assigning Pixels to Object Boxes

Jonas Uhrig[1,2], Eike Rehder[1], Björn Fröhlich[3], Uwe Franke[1], and Thomas Brox[2]

***Abstract*— The task of semantic instance segmentation has gained a large interest within academia as well as industry, especially in the context of autonomous driving. While several published approaches achieve very strong results, only few of them achieve frame rates that are sufficient for the automotive domain. We present an approach that achieves competitive results on the Cityscapes [1] and KITTI [2] datasets, while being twice as fast as any other existing approach. Our method relies on a single fully-convolutional network (FCN [3]) predicting object bounding boxes, as well as pixel-wise semantic object classes and an offset vector pointing to corresponding object centers. Using those outputs, we present an efficient and simple post-processing that assigns each object pixel to its best matching object detection, resulting in an instance segmentation obtained at real-time speeds.**

I. INTRODUCTION

Although semantic instance segmentation has experienced large progress within the last few years, it still remains a very challenging task within the vision community. Instance segmentation describes the problem of predicting a set of objects with pixel-accurate outlines, opposed to object detection, where axis-aligned bounding boxes give a rather coarse object localization. It combines the task of semantic segmentation, where each pixel of a given image has to be labeled with a semantic class (*e.g.* road, car, person, ...), with the task of object detection. Very recently, Kirillov et al. [4] proposed to further refine instance segmentation to *Panoptic Segmentation*, where predicted object masks may not overlap, while additionally requiring a full and consistent semantic segmentation of the scene.

One of many applications of semantic instance segmentation is the field of autonomous driving. Therefore, several datasets were created with automotive applications in mind, *e.g.* Cityscapes [1] and KITTI [2] (extended to pixel-level accuracy by [5] and [6]), as well as Mapillary Vistas [7]. This progress is driven by the need to accurately determine the location of individual objects, where axis-aligned boxes might be too coarse. This is most often the case for highly occluded objects and non-rectangular objects, where the actual object sometimes covers as little as 30% of the area of its surrounding bounding box.

The key challenge in semantic instance segmentation is the combination of high-level object knowledge with low-level pixel information. Approaches relying mainly on pixel information (*e.g.* as output of an FCN) usually apply complex post-processing like graphical models [6], [8], clustering [9], [10] or template matching [11], [12] to gain high-level object knowledge. On the other hand, approaches relying mainly on object detection must employ additional (sub-) networks to achieve a pixel-accurate object mask [13]. Our approach represents a balanced fusion of object and pixel knowledge, which produces accurate instance segmentation with an efficient single FCN forward-pass and a single image pass post-processing.

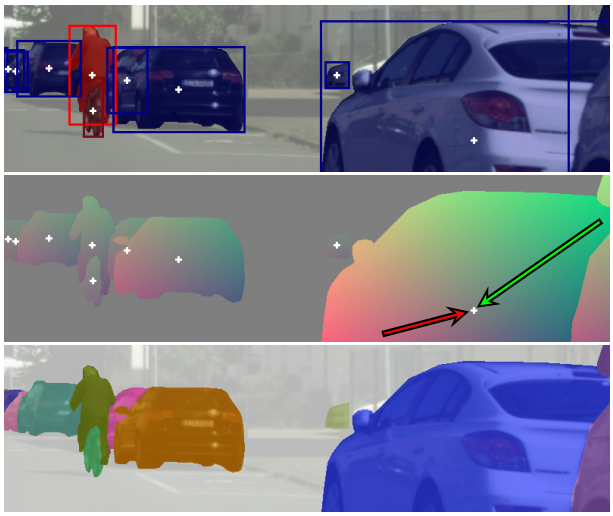

Fig. 1. **Overview:** Our method predicts object bounding boxes and semantic segmentation (top), as well as pixel-wise offset vectors (center) to corresponding object centers (white crosses) - color encoding such that red pixels point to the top right, green pixels point to the bottom left. This allows us to efficiently segment all object instances in the scene (bottom).

II. RELATED WORK

When analyzing the broad range of modern instance segmentation approaches, we find three kinds of approaches: The first tries to integrate the task of instance segmentation into the task of object detection, where Mask R-CNN [13] is the most recent and successful representative. The second combines several deep neural networks, typically one for high-level, object-based information and another for low-level, pixel-based information. The third relies mostly on pixel-features generated by a single FCN forward pass, additionally employing some post-processing to gain high-level object knowledge.

Very recently, Mask R-CNN by He et al. [13] introduced a powerful way of predicting object bounding boxes along with a pixel-accurate mask for each predicted object, based on the

[1]Image Understanding Group, Daimler AG R&D, Stuttgart, Germany {firstname.lastname}@daimler.com
[2]Department of Computer Science, University of Freiburg, Germany
[3]JENOPTIK Optical Systems GmbH, Jena, Germany

Faster R-CNN framework [14]. With a special ROI-Align operation, feature maps from an object detector network are reused and fused with an additional smaller network to produce a foreground-mask for each detected bounding box. Similarly, Straight-to-Shapes [15] predicts object box parameters (class and box dimensions) along with a compact learned mask representation in the shape of a 1×1 feature vector. A decoder network recovers pixel-level mask information from the predicted compact vector. While Mask R-CNN currently achieves highest performance on instance segmentation, it requires relatively long execution times due to the per-proposal computations and its deep base network. Straight-to-Shapes represents a very lean and efficient design, however it does not reach sufficient performance in terms of instance segmentation.

Even though the two methods above combine several (sub-) networks to address instance segmentation, the following approaches employ rather complex network architectures and post-processing: Romera-Paredes et al. [16], as well as Ren et al. [17] suggest to use a recurrent network on top of a feature-extractor base network to iteratively predict object instances. The Sequential Grouping Network by Liu et al. [18] present an approach predicting horizontal and vertical pixel-based instance boundaries with one network and grouping them with another. Fully Convolutional Instance Segmentation [19], [20] is based on Faster R-CNN, predicting multiple instance-sensitive score maps per proposal to achieve high-quality object masks. Boundary-aware Instance Segmentation [21] presents an extension for Faster R-CNN predicting per-pixel distances to proposal centers in order to allow some pixels outside of the predicted object box to still be assigned to the respective box. The Multi-task Network Cascade [22] describes another adaptation of Faster R-CNN to sequentially predict object proposals, their masks and eventually their class, reusing extracted features of previous tasks. MaskLab [23] describes concurrent work to our approach, using Faster R-CNN as object detector and an additional per-proposal pixel-wise class prediction describing direction and distance to the object center. In contrast, we employ a fast Single-Shot Detector and predict center-offsets as regression on the full image at once.

The following approaches all rely on a single FCN to predict strong pixel-based features that allow to form object instances with pixel-accurate outlines in a consecutive post-processing step. Using efficient base architectures, the runtime of those approaches highly depends on the employed post-processing step, which tends to be rather slow. Also, as those approaches do not explicitly incorporate high-level object knowledge, they tend to degenerate on large objects and on scenes with large and complex occlusions. Many works use loss variants of typical metric learning problems in order to address instance segmentation by predicting embedding vectors that differ across objects [9], [10], [24], [25]. They achieve high-level object representations by applying a clustering on the high-dimensional feature embeddings or predict seed points from where instances are grown via similarity. The method by Neven et al. [24] represents a particularly efficient solution for instance segmentation, however, they report performance only on one of eight object classes of the Cityscapes dataset. Also, to achieve fast execution, they rely on an image downsampling, which can be problematic for small objects (*i.e.* distant or highly occluded). Another row of work employs classical computer vision post-processing on top of suitable FCN features: Template matching [11], [12], watershed [26] or connected component analysis [27], multicut [28], graph optimization [29], or graphical models [6], [8], [30]. The Proposal Free Network [31] predicts object classes and box parameters for a fixed feature-grid resolution in a single shot (*c.f.* YOLO [32]), while additionally regressing an overall number of visible objects per class - using an off-the-shelf clustering on the box parameters, they can extract object instances.

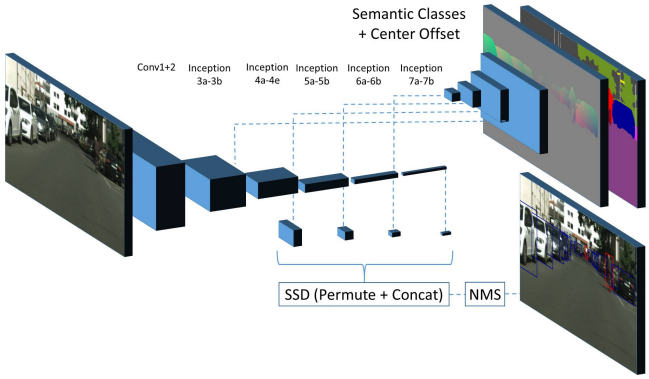

Fig. 2. **Our overall network architecture:** A single FCN in GoogLeNet-v1 style [33] predicts per-pixel semantics and center-offset via deconvolutions with skip-connections [3] and boxes using an SSD head [34].

In contrast to previous works, we employ an efficient single-shot detector together with an efficient and simple pixel-to-box assignment, which is based on a per-pixel prediction of semantic classes and offset vectors to object centers. All those features are predicted with a single network that is based on the efficient GoogLeNet-v1 architecture [33], using SSD [34] as object detection head. This allows our method to run on frame rates sufficient for autonomous driving: over 10Hz on Cityscapes' 2MP images and 35Hz on KITTI images (approximately 0.5MP) - with a standard TensorFlow [35] implementation on an Nvidia Titan Xp. Also, we present several adaptations of the original SSD [34], lying in the network architecture, choice of prior boxes, and loss formulation.

III. METHOD

A. Network Structure

To predict all inputs for our post-processing (bounding boxes and per-pixel semantic class as well as offset vectors to object centers), we use an adaptation of the GoogLeNet-v1 Inception architecture [33], *c.f.* Fig. 2. We find this architecture to be very efficient, allowing real-time execution on 2MP camera images with very high precision in semantic segmentation performance, *c.f.* extensive studies in [36].

For the SSD head of our network, we remove the original average pooling after GoogLeNet's inception 5b module and replace it with an ordinary 2-strided max pooling. Then, instead of adding (partially strided) Conv6 to Conv11_2, as suggested in [34], we add two additional inception modules, *i.e.* 6a and 6b, with same parameters as inception 5a and 5b but on stride 64 instead of stride 32. To achieve even larger receptive fields for large objects such as trucks, trains, and buses that might fill the whole 2MP width of 2048 pixels, we add another 2-strided max pooling and inception modules 7a and 7b, again with parameters of 5a and 5b. For predicting actual box parameters and box classes, we add 1×1 convolutional branches from following inception module outputs: 4e, 5b, 6b, and 7b. We recursively compute the theoretical receptive field of those layers using

$$\mathrm{RF}_{\mathrm{out}} = (\mathrm{RF}_{\mathrm{in}} - 1) * s + k$$

where RF describes the receptive field of input and output, s is a layer's stride and k describes its kernel size: 427 pixels squared for inception 4e, 715 for 5b, 1291 for 6b, and eventually 2443 for inception 7b. However, from studies of effective receptive fields [37], we know that actual receptive fields after training can be significantly smaller. Therefore, we require the theoretical receptive field to be twice the maximal width or height of a prior box in order to assign it to a specific layer.

For our pixel-wise predictions, namely semantic classes and center-offset vectors, we employ the strategy presented in [3]: We use skip connections from corresponding inception-module outputs, consisting of 1×1 convolutions and an element-wise addition, as well as deconvolutions to sequentially upscale low-resolution feature maps. We find that using 2 filters for center-offsets, and $n + 1$ filters for n object classes and a background class for the upsampling part of the network, as well as a starting point at inception 6b to give the best runtime vs. performance trade-off.

B. Loss Formulation

To weight the four different tasks that our network shall solve (semantics, offsets, as well as object location and detection scores), we use the approach presented by Kendall et al. [38] to learn task uncertainties σ:

$$\begin{aligned} L_{\mathrm{total}} \quad = \quad & \frac{1}{\sigma^2_{\mathrm{semantics}}} L_{\mathrm{semantics}} + \log \sigma_{\mathrm{semantics}} \\ & + \frac{1}{2\sigma^2_{\mathrm{offsets}}} L_{\mathrm{offsets}} + \log \sigma_{\mathrm{offsets}} \\ & + \frac{1}{2\sigma^2_{\mathrm{SSD_{box}}}} L_{\mathrm{SSD_{box}}} + \log \sigma_{\mathrm{SSD_{box}}} \\ & + \frac{1}{\sigma^2_{\mathrm{SSD_{class}}}} L_{\mathrm{SSD_{class}}} + \log \sigma_{\mathrm{SSD_{class}}} \end{aligned} \tag{1}$$

For semantic segmentation, we use a standard cross-entropy loss as $L_{\mathrm{semantics}}$. As loss for center offset vectors L_{offsets}, we use an L2 regression loss. Both are normalized over the number of valid pixels. $L_{\mathrm{SSD_{box}}}$ describes our regression loss for SSD box parameters (parametrized as $\mathrm{x_{min}}$, $\mathrm{y_{min}}$, $\mathrm{x_{max}}$, $\mathrm{y_{max}}$), which is implemented as an L2 loss. For our box classification loss $L_{\mathrm{SSD_{class}}}$, we use Focal Loss [39] to cope for the extreme class imbalance between foreground and background labels - in Sec. IV-A, we evaluate several parameterizations for the necessary γ parameter.

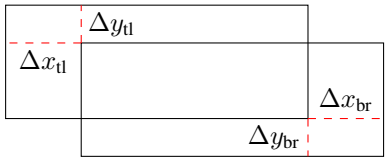

Fig. 3. **Relative Change Metric:** Using the top left (tl) and bottom right (br) corners of two boxes (typically one prior box and one GT box), we compute the relative change as described in Equation 2.

C. SSD Adaptations

We identified a weakness in the original SSD implementation [34] that causes only few GT boxes being associated during training: A predefined threshold is used in order to state whether a predicted bounding box and an annotated ground truth (GT) box match, which is necessary to punish false-positives and false-negatives. This threshold is defined as 50% in terms of intersection over union (IoU) of each possible template box and the GT boxes. For the training split of Cityscapes [1], we find that only about 40% of all GT boxes have an overlap larger than this threshold with any existing prior box. Due to a lack of sufficient supervision, this causes rather bad object detection performance, especially for small objects, *c.f.* Sec. IV-A. By reducing the IoU threshold, it is possible to increase the number of prior boxes that get matched with a GT box, however, this also causes rather dissimilar prior boxes to match with a given GT box. Overall, we find that reducing the IoU threshold to less than 50% causes even worse object detection performance.

To overcome this problem, we propose a novel metric to find well matching prior boxes: We use relative box parameter changes instead of IoU to both overcome issues on small objects and low coverage rates. For two boxes, one prior box b_{prior} with parameters $(x_{\mathrm{min}}, y_{\mathrm{min}}, x_{\mathrm{max}}, y_{\mathrm{max}})$ and one annotated ground truth box b_{GT} with width $w_{\mathrm{GT}} = x_{\mathrm{max}} - x_{\mathrm{min}}$ and height $h_{\mathrm{GT}} = y_{\mathrm{max}} - y_{\mathrm{min}}$, we compute the relative change as follows:

$$\mathrm{d_{change}} = \sqrt{\frac{\Delta y_{\mathrm{tl}}}{h_{\mathrm{GT}}}^2 + \frac{\Delta x_{\mathrm{tl}}}{w_{\mathrm{GT}}}^2 + \frac{\Delta y_{\mathrm{br}}}{h_{\mathrm{GT}}}^2 + \frac{\Delta x_{\mathrm{br}}}{w_{\mathrm{GT}}}^2} \tag{2}$$

Where Δy is the absolute difference between y parameters of both boxes (y_{min} for the top left (tl) box corner, y_{max} for the bottom right (br) corner), and Δx analogously for x parameters, *c.f.* Fig. 3.

We find our change metric more intuitive than an IoU-based matching, as box regression is commonly trained using corner offsets, not to optimize IoU overlaps. And indeed, having loss and matching in the same space seems to facilitate the training process, as we experience much smoother convergence behavior when comparing change-based matching versus IoU-based matching.

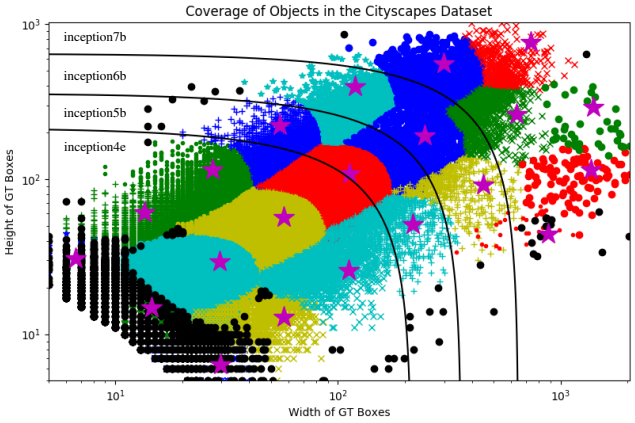

Fig. 4. **Coverage of Prior Boxes:** Our 21 selected prior box dimensions (purple stars) densely cover most Cityscapes [1] objects (black dots). Colored areas have $d_{change} \leq 0.7$ to any prior box, *c.f.* Equation 2. Black lines represent upper bounds of object dimensions according to half the theoretical receptive field of each SSD output layer.

In addition to the IoU-based matching, we find the choice of prior boxes as suggested in [34] to be rather inefficient in terms of number of GT boxes within the matching threshold vs. number of created prior boxes. We therefore tried to find an optimal set of prior box dimensions by clustering, as suggested by Redmon et al. [40]. However, we find that mostly very small prior boxes are generated, while some rare large objects are ignored. This effect occurs both when clustering in IoU distance as well as relative change, however, not as drastically. Therefore, we manually select a set of 21 prior boxes to densely cover the wide range of object dimensions with a minimal amount of priors and assign them to SSD output layers respecting receptive field sizes, *c.f.* Fig. 4.

D. Pixel-to-Box Assignment

In the following section, we describe how to generate a semantic instance segmentation using the three proposed network outputs (pixel-wise semantic class and center-offset vectors, as well as a set of predicted object boxes with detection scores).

The very first step, based solely on the detected object boxes lies in a common non-maximum-suppression (NMS), which is implemented in the object detection tools of TensorFlow [35]: We take the argmax non-background detection for each SSD grid cell (each prior box location can only predict a single object class), typically resulting in a set of predictions for a single object. Then, the prediction scores are sorted in descending order, where all predictions with a lower score are removed if they have an IoU overlap larger than a given threshold (usually 50%) with any other prediction with a higher score.

Followingly, we find the best matching object box for each pixel by computing the minimal distance between the pixel-wise center prediction and the predicted bounding boxes: To determine the estimated per-pixel object center location, we add each pixel's location to the predicted center-offset vectors. Next, we extract all foreground pixel indices and map all predicted semantic labels to corresponding box class labels, if necessary. Using those indices, we can rearrange each pixel's location, its estimated object center and semantic class in vector form, which allows further processing in efficient-to-compute GPU tensor operations: We use an outer addition, similar to an outer product, to subtract a $[n, 1]$ shaped vector b from a $[1, m]$ shaped vector p. This generates a $[n, m]$ shaped matrix D, where $D_{i,j} = b_i - p_j$. For the following, n will be the number of non-maximum-suppressed boxes and m will be the number of non-background pixels. We find the best matching box for each single pixel by incorporating several distances: The most obvious distance is that between a pixel's predicted object center and the actual center of a predicted box, *c.f.* d_{off} in Fig. 5. We reformulate this pair-wise distance computation as a matrix multiplication in order to efficiently compute many distances in parallel. Also, to punish pixels that lie outside of boxes and to assign them to such boxes less likely, we add d_{out} which is each pixel's distance to the closest point on the predicted box, and zero if it lies inside the box. To heavily punish label inconsistencies, *e.g.* to not assign pedestrian pixels to predicted vehicle boxes, we add a sufficiently large number to the overall distance whenever pair-wise labels mismatch. Also, we find that even after successful NMS, there might still be multiple detections on the same object. However, due to the previously described behavior of NMS, those additional detections either have a rather bad IoU overlap with the actual object or a rather low detection score. Therefore, we add a small additional distance (relative to image dimensions) to each box-pixel pair for detection scores lower than 100%.

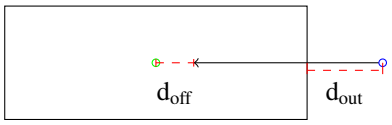

Fig. 5. **Pixel-to-Box Assignment:** Each pixel (blue circle) predicts an offset vector (black arrow) towards an object center (black rectangle with green circle). If the pixel lies outside of a predicted box, it is punished with the distance to the closest box boundary d_{out}. Additionally, the difference between offset prediction and box center is punished with d_{off}.

Eventually, after adding all additional distance terms to our $D_{i,j}$, we can easily extract the index of the *closest* box prediction for each non-background pixel using an argmin over all rows of D. As we stored the locations of all non-background pixels, we can now simply assign those pixels with the extracted argmin indices as instance IDs. For evaluating this instance ID image on most modern benchmarks, we still need to generate a prediction score and a class label for each instance. For prediction scores, we can simply reuse the detection scores of respective selected box predictions. For class labels, we can either use the box class label if we have the same classes in box prediction and semantic segmentation, or use the most frequently appearing semantic class among all pixels that got assigned to a specific box.

TABLE I

INSTANCE SEGMENTATION ON CITYSCAPES VALIDATION

Variant	AP (instances)	$AP^{50\%}$ (instances)	$mAP^{50\%}$ (objects)	IoU (semantics)
IoU				
50% FG/BG	6.4%	12.4%	34.8%	58.2%
FL ($\gamma = 2$)	8.1%	15.3%	25.2%	63.1%
FL ($\gamma = 5$)	8.4%	14.8%	24.7%	63.6%
Change				
50% FG/BG	12.4%	24.2%	44.5%	65.5%
FL ($\gamma = 2$)	**17.4%**	**33.1%**	**49.2%**	**66.2%**
FL ($\gamma = 5$)	14.5%	28.9%	47.2%	65.5%

TABLE II

INSTANCE SEGMENTATION ON KITTI VALIDATION

Variant	AP (instances)	InsF1 (instances)	$mAP^{50\%}$ (objects)	IoU (semantics)
50% FG/BG	47.0%	75.1%	**88.5%**	93.2%
FL ($\gamma = 2$)	46.0%	**87.3%**	84.5%	93.9%
FL ($\gamma = 5$)	**47.1%**	84.2%	82.7%	**94.1%**

This operation can be formulated as the argmax of an outer product of one-hot encoded vectors of box and pixel classes. The actual maximum value over all columns of this one-hot vector matrix multiplication gives the number of pixels that were assigned to an individual box. This information might be used to further filter out instances where only very small portions of a predicted bounding box represent an object, which might be an indicator for false-positives.

Overall, all presented steps can be easily formulated in a TensorFlow computation graph, which allows GPU computation of the complete pixel-to-box assignment. We experience average execution times of approximately 100 milliseconds on Cityscapes images (2MP), and 30 milliseconds in KITTI images (approx. 0.5MP) with plain TensorFlow 1.4 [35] and CUDNN version 6, using an Nvidia Titan Xp.

IV. EXPERIMENTS

In the following, we evaluate our approach on two challenging datasets, namely Cityscapes [1] and KITTI [2], [5], [6] using suggested instance-aware metrics. We extract object bounding boxes from provided instance segmentation annotations, *i.e.* minimal and maximal pixel locations for each object instance. We train our network in the TensorFlow framework [35] with the Adam optimizer [41], using standard momentum settings of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, as well as a learning rate of 10^{-4}.

A. Ablation Studies

As described in Sec. III-C, we implement several adaptations of the original SSD approach [34]. We find a significant performance boost from using our suggested change-based metric (threshold of 0.7) compared to the traditional IoU-based metric (threshold 0.5) to find matches between predictions and GT boxes, *c.f.* "IoU" and "Change" sections in Table I. While semantic segmentation performance stays at a same level for all experiments (approx. $60\% - 65\%$), we find large differences in the object mean average precision:

TABLE III

TEST SET EVALUATION FOR CITYSCAPES. EXECUTION TIMES, IF NOT STATED BY THE AUTHORS, ARE GUESSED IN FAVOR OF THE APPROACH.

Method	AP	$AP^{50\%}$	FPS
Mask-R-CNN (COCO) [13]	**32.0%**	**58.1%**	< 1
Mask-R-CNN [13]	26.2%	49.9%	< 1
SGN [18]	25.0%	44.9%	0.6
Dynamic Net [8]	20.0%	38.8%	< 3
DWT [26]	19.4%	35.3%	< 3
Discriminative Loss [9]	17.5%	35.9%	5
BAIS [21]	17.4%	36.7%	≤ 1
Instance Cut [28]	13.0%	27.9%	< 1
Foveal Vision [12]	12.5%	25.2%	< 3
Graph Decomposition [29]	9.8%	23.2%	≤ 1
Recurrent Attention [17]	9.5%	18.9%	< 3
Pixel-Encoding [11]	8.9%	21.1%	< 3
R-CNN + MCG [1]	4.6%	12.9%	0.02
Deep Contours [43]	2.3%	3.6%	5
Ours (FL, $\gamma = 2$)	13.1%	27.2%	**10.9**

Using the IoU-based matching, trained networks tend to miss many small objects, which is not the case for the change-based matching. We did not investigate correlations with object detection approaches that specifically focus on small objects, such as Feature Pyramid Network [42], which is left for future work.

When fixing the used metric, *e.g.* to relative change, we find (less significant) performance boosts from using Focal Loss (FL) [39] over traditional weighting schemes for foreground (FG) and background (BG), *e.g.* 50% each. Experimentally, we find a γ of 2 to achieve highest detection scores (mAP of 49.2%) as well as highest instance segmentation scores (AP of 17.4%) on the Cityscapes validation split. Similar trends are visible on the KITTI validation split, *c.f.* Table II, however not as clear as for Cityscapes. Even though $\gamma = 5$ achieves slightly better AP performance, we chose $\gamma = 2$ as InsF1 and object detection performance are significantly better.

B. Segmentation Performance vs. Runtime

We compare our instance segmentation performance with other baseline approaches on the test set of Cityscapes, *c.f.* Table III, as well as the KITTI test split, *c.f.* Table IV. While we achieve a new state-of-the-art on KITTI on all metrics but instance-level precision, we fall behind on Cityscapes. We experience a significant performance drop from 17.4% AP on Cityscapes validation down to 13.1% on the test set. We find no other reason than higher difficulty, as most related work experiences a similar performance drop. Nevertheless, we find our method in the middle in terms of average precision with clearly faster execution times (twice as fast as the second fastest). Brabandere et al. [9] achieve 17.5% AP with 5 FPS, however to perform such fast execution times, they downsample images to a resolution of 768×384, which is almost an eighth of the original 2MP. For methods that do not state execution times on the official benchmark or in their publications, we guess in favor of the method using the employed network base architecture (*e.g.* ResNet-50 [44], VGG [45], or Dilation-10 [46]) and execution times reported

Fig. 6. **Result Samples:** Result on Cityscapes (top) and KITTI (bottom), using the Cityscapes color encoding for semantic segmentation and random colors for instance segmentation. Most remaining failure cases are based on missed or double object boxes or erroneous semantics. Best viewed on screen.

TABLE IV
TEST SET EVALUATION FOR KITTI - ALL METRICS IN PERCENT, WHERE HIGHER IS BETTER

Method	**AP** (instances)	**AP**$^{50\%}$ (instances)	**MWCov** (instances)	**InsPr** (instances)	**InsRe** (instances)	**InsF1** (instances)	**IoU** (semantics)	**AP**$^{50\%}$ (objects)
Depth Ordering [30]	-	-	70.9	40.2	45.9	42.8	77.3	-
Deep MRF [6]	-	-	74.1	70.9	53.7	61.1	78.5	-
Graph Decomposition [29]	43.6	71.4	-	69.2	76.5	72.7	83.9	-
Pixel-Encoding [11]	41.6	69.1	79.7	**86.3**	74.1	79.7	84.1	-
Recurrent Attention [17]	-	-	80.0	-	-	-	87.4	-
Ours (FL, $\gamma = 2$)	**44.6**	**72.5**	**89.0**	84.6	**80.7**	**82.6**	**93.5**	**84.2**

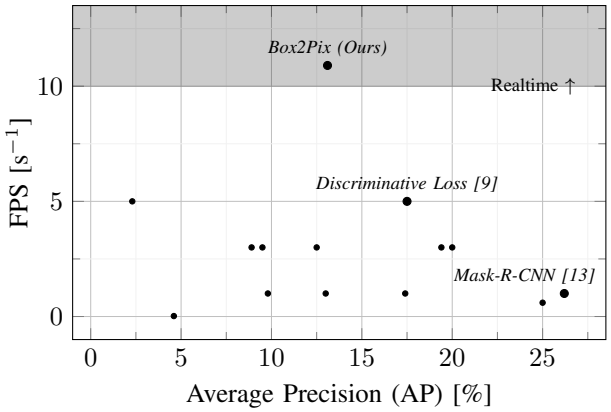

Fig. 7. **Runtime vs. Performance:** Current submissions on the Cityscapes test benchmark [1], using only Cityscapes training data. Execution times, if not stated in the original paper, are guessed in favor of the method.

in [36]. Other methods, such as Mask-R-CNN, report 5 FPS on 0.4MP images, which extrapolates optimistically to 1 FPS on 2MP Cityscapes images, ignoring additional time for the increased number of objects. See Fig. 7 for a visualization of runtime vs. performance numbers for approaches trained on Cityscapes.

Huang et al. [47] find that SSD is the fastest under the good performing deep object detectors, *i.e.* one of the most efficient base architectures. By further adjusting the base architecture to a GoogLeNet-v1 [33], we measure the following execution times: For a full 2MP image, the network (including SSD and deconvolution heads) requires approximately 70 milliseconds on an Nvidia Titan Xp GPU for inference. The non-maximum-suppression as implemented in TensorFlow [35] requires in average 10 ms, and our pixel-to-box assignment requires another 20 ms, resulting on average in 100 milliseconds for the complete instance segmentation. For 0.5MP large images, the overall computation lasts 30 ms with 20 for the network and 8.5 for NMS plus our assignment. When implemented efficiently on a GPU, execution times of 10 ms for NMS on Cityscapes images go down to 2 ms, resulting in an overall runtime of 92 ms. In future work, instead of densely computing distances between all pixels and all bounding boxes in the image, one could further reduce runtime by only computing distances for pixels in the neighborhood of a detected object box.

V. CONCLUSION

We present an approach that is capable of segmenting object instances in real time, even on 2MP large images. We achieve competitive performance on automotive datasets such as Cityscapes [1] and KITTI [2], being twice as fast as the second fastest competitive approach, without relying on image downsampling. This is possible by employing the efficient state-of-the-art base network GoogLeNet-v1 [33] and SSD [34] as object detection head. Additionally, we propose to extend the GoogLeNet-v1 architecture by several inception modules for an efficient deep object detection network. Also, we find a novel metric to define positive matches in the SSD loss computation (relative box parameter changes), which produces superior object detection and instance segmentation performance compared to IoU-based matching approaches.

In future work, we plan to further boost our object detection performance by incorporating special approaches for small and highly occluded objects, which are our main failure case. Also, with the availability of new challenges incorporating *Panoptic Segmentation* [4], we plan to extend our approach to predict complete semantic segmentation instead of only predicting object and background classes.

REFERENCES

[1] 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 *CVPR*, 2016. 1, 3, 4, 5, 7

[2] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *CVPR*, 2012. 1, 5, 7

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015. 1, 2, 3

[4] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *arXiv:1801.00868 [cs.CV]*, 2018. 1, 7

[5] L. C. Chen, S. Fidler, and R. Urtasun, "Beat the MTurkers: Automatic image labeling from weak 3d supervision," in *CVPR*, 2014. 1, 5

[6] Z. Zhang, S. Fidler, and R. Urtasun, "Instance-level segmentation with deep densely connected MRFs," in *CVPR*, 2016. 1, 2, 5, 7

[7] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *International Conference on Computer Vision (ICCV)*, 2017. 1

[8] A. Arnab and P. H. S. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," in *CVPR*, July 2017. 1, 2, 5

[9] B. D. Brabandere, D. Neven, and L. V. Gool, "Semantic instance segmentation with a discriminative loss function," in *CVPR Workshop*, 2017. 1, 2, 5, 7

[10] S. Kong and C. Fowlkes, "Recurrent pixel embedding for instance grouping," in *arxiv:1712.08273 [cs.CV]*, 2017. 1, 2

[11] J. Uhrig, M. Cordts, U. Franke, and T. Brox, "Pixel-level encoding and depth layering for instance-level semantic segmentation," in *GCPR*, 2016. 1, 2, 5, 7

[12] B. Ortelt, C. Herrmann, D. Willersinn, and J. Beyerer, "Foveal vision for instance segmentation of road images," in *VISAPP*, 2018. 1, 2, 5

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017. 1, 5, 7

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015. 2

[15] S. Jetley, M. Sapienza, S. Golodetz, and P. Torr, "Straight to shapes: Real-time detection of encoded shapes," in *arXiv:1611.07932 [cs.CV]*, 2016. 2

[16] B. Romera-Paredes and P. Torr, "Recurrent instance segmentation," in *arXiv:1511.08250v2 [cs.CV]*, 2015. 2

[17] M. Ren and R. S. Zemel, "End-to-end instance segmentation with recurrent attention," in *CVPR*, 2017. 2, 5, 7

[18] S. Liu, J. Jia, S. Fidler, and R. Urtasun, "SGN: Sequential grouping networks for instance segmentation," in *ICCV*, Oct 2017. 2, 5

[19] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *CVPR*, 2017. 2

[20] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," in *ECCV*, 2016. 2

[21] Z. Hayder, X. He, and M. Salzmann, "Boundary-aware instance segmentation," in *CVPR*, 2017. 2, 5

[22] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *CVPR*, 2016. 2

[23] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "MaskLab: Instance segmentation by refining object detection with semantic and direction features," in *arXiv:1712.04837 [cs.CV]*, 2018. 2

[24] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, "Fast scene understanding for autonomous driving," in *IV Symposium Workshop*, 2017. 2

[25] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy, "Semantic instance segmentation via deep metric learning," in *arXiv:1703.10277 [cs.CV]*, 2017. 2

[26] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *CVPR*, 2017. 2, 5

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015. 2

[28] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, "Instancecut: from edges to instances with multicut," in *CVPR*, 2017. 2, 5

[29] E. Levinkov, J. Uhrig, S. Tang, M. Omran, E. Insafutdinov, A. Kirillov, C. Rother, T. Brox, B. Schiele, and B. Andres, "Joint graph decomposition and node labeling: Problem, algorithms, applications," in *CVPR*, 2017. 2, 5, 7

[30] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun, "Monocular object instance segmentation and depth ordering with cnns," in *ICCV*, 2015. 2, 7

[31] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan, "Proposal-free network for instance-level object segmentation," in *arXiv:1509.02636v2 [cs.CV]*, 2015. 2

[32] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016. 2

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015. 2, 7

[34] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016. 2, 3, 4, 5, 7

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *OSDI*, 2016. 2, 4, 5, 7

[36] M. Cordts, "Understanding Cityscapes: Efficient Urban Semantic Scene Understanding," Ph.D. dissertation, Technische Universität, Darmstadt, 2017. 2, 7

[37] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *NIPS*, 2016. 3

[38] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *arXiv:1705.07115 [cs.CV]*, 2017. 3

[39] T.-Y. Lin, P. Goyal, R. Girshick, K. He, , and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017. 3, 5

[40] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *CVPR*, July 2017. 4

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv:1412.6980 [cs.CV]*, 2014. 5

[42] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *arXiv:1612.03144 [cs.CV]*, 2017. 5

[43] J. van den Brand, M. Ochs, and R. Mester, "Instance-level segmentation of vehicles by deep contours," in *ACCV Workshop*, 2016. 5

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016. 5

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *arXiv:1409.1556 [cs.CV]*, 2014. 5

[46] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016. 5

[47] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *arXiv:1611.10012 [cs.CV]*, 2016. 7