Fast Instance and Semantic Segmentation Exploiting Local Connectivity, Metric Learning, and One-Shot Detection for Robotics

Andres Milioto Leonard Mandtler Cyrill Stachniss

***Abstract*— Semantic scene understanding is important for autonomous robots that aim to navigate dynamic environments, manipulate objects, or interact with humans in a natural way. In this paper, we address the problem of jointly performing semantic segmentation as well as instance segmentation in an online fashion, so that autonomous robots can use this information on-the-go and without sacrificing accuracy. We achieve this by exploiting a local connectivity prior of objects in the real world and a multi-task convolutional neural network architecture. The network identifies the individual object instances and their classes without region proposals or pre-segmentation of the images into individual classes. We implemented and thoroughly evaluated our approach, and our experiments suggest that our method can be used to accurately segment instance masks of objects and identify their class in an online fashion.**

I. Introduction

Perception is one of the main building blocks for robotic applications that need interaction with real-world, dynamic environments such as city roads or domestic environments. It is important for robots to operate safely and in an situation-dependent manner. Semantic information about the environment in the form of object detection, semantic segmentation, and instance segmentation have been exploited for many robotics tasks such as mapping [9], [14], [19], [32], visual place recognition [11], and manipulation [4], [18], [21], [31].

One of three different types of approaches is typically used in this context: semantic segmentation, object detection, and instance segmentation. *Semantic segmentation* (see Fig. 1, middle) conveniently provides a class label for each pixel of an image, and therefore enables applications that require accurate object masks such as removal of dynamic objects for visual odometry. However, it does not provide an association of the pixels to an object instance. This is inconvenient for robotic tasks where the location of each object is of interest, for example, to be used as a visual landmark, for object manipulation, or collision avoidance. Furthermore, performing semantic segmentation is often computationally demanding. This is especially true if using high resolution images, which is often desired for high accuracy and handling objects at a far distance. On the other side of the spectrum, *object detection* provides a class label for each object instance jointly with the regression of the image coordinates of a bounding box that encloses the object. This approach is popular as it has the advantage of fast inference. Unfortunately, object detection does not provide an object mask, and the correlation between the object boundaries and the bounding boxes can be hard to infer.

All authors are with the University of Bonn, Germany. This work has partly been supported by the German Research Foundation under Germanys Excellence Strategy, EXC-2070 - 390732324 (PhenoRob), and NVIDIA Corporation.

Fig. 1. Desired workflow. Top: Input RGB image. Middle: Semantic segmentation mask. Bottom: Instance segmentation mask.

The holy grail for semantic scene understanding in robotics is *instance segmentation* (see Fig. 1, bottom), as it not only identifies each individual object instance, but also provides an accurate segmentation mask and thus enables a further, task-specific, analysis of the image content.

The main contribution of this paper is a novel approach that in parallel performs instance segmentation and semantic segmentation, both in an efficient and effective manner for robotics. We propose a convolutional neural network (CNN) architecture that uses superpixel summarization of locally connected regions of an image, and combines object detection and a metric learning pipeline to speed up joint semantic and instance segmentation without sacrificing accuracy. Our CNN predicts, for each output superpixel, the probability of it being an instance center, a high-dimensional embedding from a metric learning loss, and a softmax probability for the semantic segmentation task. This approach allows us to perform fast upsampling without sacrificing mask accuracy, while at the same time speeding up the clustering of the embeddings to obtain an individual instance mask for each object in the scene.

II. Related Work

Several methods for instance segmentation have been proposed over the last years, most of them based on some CNN backbone for feature extraction but using different types of decoders. Such approaches can be grouped according to how they exploit different decoder architectures.

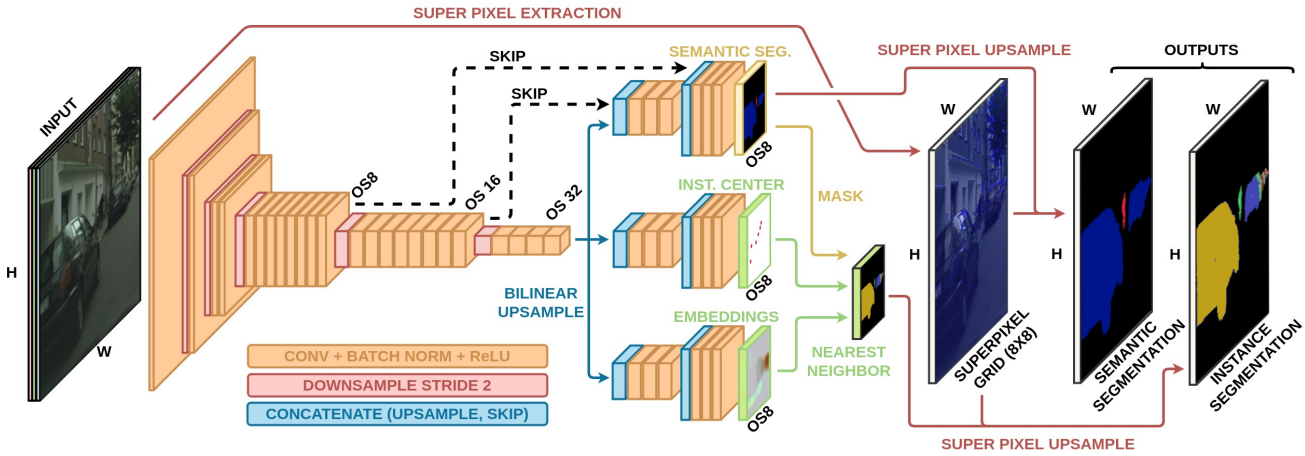

Fig. 2. Our architecture. The different resolution features are skipped to all decoders in their respective output stride (OS). Each decoder predicts a volume of lower resolution than the input, in this case of OS8, reducing the number of pixels to cluster by a factor of 64. After the decoders predict the semantic mask, and the centers and embeddings are joined into individual instances, the superpixels are used to upsample the results. The encoder shown is Darknet53 [25].

One group of approaches works with *region proposals* coming from an object detection pipeline and a posterior mask segmentation between foreground and background. This has the advantage that these approaches can deal with an arbitrary object number and usually provide accurate results. State-of-the-art examples are Mask-RCNN [12] or PANet [17]. Such type of pipeline is particularly useful in application-specific use cases where obtaining region proposals is easy, for example in [20]. However, there are two main disadvantages of these types of proposal-based, two-stage pipelines. First, they are usually slow and second, the relationship between the object mask and its bounding box may not be straight-forward. This may render the computation of the object mask hard in some cases. Furthermore, most of these approaches infer the segmentation of the foreground class in each bounding box proposal in a reduced resolution, relying on bilinear upsampling to map it back to the original image size. This sacrifices the accuracy of the object boundaries.

The second group of approaches relies on *recurrent neural networks* and *attention mechanisms* [27], [29]. They take a single image as an input and output instance masks, one by one in a counting-like manner. This is sometimes reported as biologically inspired, but these approaches do not offer the best accuracy and runtime.

The last group of approaches to instance segmentation is based on *metric learning* strategies [3], [5], [10], [22], [24]. They rely on predicting a discriminative high-dimensional embedding for each pixel, followed by a clustering of pixel embeddings into individual instances. Some methods in this context rely on a previously executed, semantic segmentation of the input into the individual semantic classes, and subsequently predict an embedding [5] or an energy function [3] that allows to separate the pixels inside each semantic class mask into individual instances. The required semantic segmentation usually comes from a computationally expensive CNN and, thus, except few examples, these approaches cannot run in real-time on robot, and their accuracy has an upper bound that is given by the performance of the pre-processing step. Alternative approaches [10], [22], try to predict both, the class labels, as well as the instance embeddings at the same time. There is, however, no concept of "objectness" in these types of pipelines, which makes the posterior clustering of the embeddings difficult and computationally demanding.

Our approach overcomes this limitation by adding the inference of the object centers confidence such as in oneshot object detection pipelines, which in turn allows us to speed up the post processing by avoiding the expensive all-to-all similarity matrix calculation. Furthermore, we exploit fast GPU-based superpixel summarization, which exploits the local connectivity of pixels in objects, and allows us to further reduce the number of computations needed to calculate the similarity matrix between the embeddings in each semantic class and the centers of clusters. This is key for achieving semantic analysis at camera frame-rate.

III. Our Approach

The main goal of our work is to obtain accurate instance segmentation masks of objects from RGB camera images in a timely manner. We propose a CNN-based algorithm that combines detection and segmentation with superpixel summarization, allowing us to obtain accurate semantic and instance labels for each pixel of an image. As our approach operates in a low dimensional feature-space grid, it is fast to run without sacrificing mask or instance performance.

Our method uses a common CNN encoder extracting features from RGB images at different resolutions, and has three separate decoder heads, see Fig. 2. Each decoder head combines and upsamples the multi-resolution features into a low-dimensional grid, which makes the processing fast. We explain these three output grids extensively, and we call the value at a certain (x, y) position of this grid a "grid element". Such an element addresses all corresponding feature values at that spatial position. The three heads predict: (i) a semantic segmentation mask which maps each grid element

to a softmax pseudo-probability distribution over the desired semantic classes; (ii) a high-dimensional embedding for each grid element, which is to be close in Euclidean similarity for elements belonging to the same instance, and distant otherwise; and (iii) the confidence of each grid element being an object center. Parallelly, the input is processed by a fast GPU-based superpixel algorithm [26] based on SLIC [1] which is able to upsample each "grid element" into the locally connected pixels in the original resolution output.

Our approach can be summarized in three main steps. First, a CNN backbone summarizes the image as a set of features of different resolution, and three task decoders upsample these features into task grids of lower resolution than the input. These decoders contain the semantic segmentation, center confidences, and embeddings respectively for each of the groups of input pixels that are mapped to it. Second, and in parallel with the CNN, a fast superpixel extraction [26] of the original image is performed resulting in a mapping used to upsample the decoder grids. Finally, a post processing is performed to map each embedding to an individual object center and extract the instances, previous to upsampling using the mappings from step two.

A. Joint Semantic and Instance Segmentation CNN

Our CNN structure is composed of four main components: (i) a fully convolutional encoder which extracts features at different resolutions for the decoders, (ii) a decoder which infers a downsampled semantic segmentation softmax distribution over the semantic classes, (iii) a decoder which infers a confidence of each superpixel being an object center, and (iv) a decoder which infers a 32-dimensional embedding for each superpixel using a discriminative metric loss. All four components are trained jointly using a weighted sum of the three task losses. In the following subsections, we introduce each of these modules in detail:

1) Encoders: we use two different convolutional backbone architectures for the multi-resolution feature extraction with different levels of descriptiveness, defined by their size and the number of layers. On the computationally expensive side of the spectrum, we use Darknet 53 [25] (DN53), which is a ResNet [13] inspired architecture and has proven to work well for the object detection task as a part of the YOLOv3 architecture. This architecture also obtains top-1 accuracy on the validations set of ImageNet-1K [8] of 77.2%, which is the current state of the art, and higher than ResNet101, which is 50% slower to run. This makes it a well-suited feature extractor for our backbone. On the other side of the descriptiveness spectrum, we use a MobilenetsV2 [30] (MNv2) encoder, which has an order of magnitude less parameters than Darknet 53 and is designed to maximize efficiency for running on mobile devices. This backbone achieves a top-1 ImageNet accuracy of 72%. This is lower than the more expensive Darknet 53, but is the current state of the art for real-time applications. Before attaching the decoders, we pretrain both backbones on ImageNet to accuracy, using the standard output stride (OS) of both backbones of 32, which means that the last layer will be downsampled 32 times from the original image size. Our framework allows the extraction of any feature resolution of the model to skip to the decoder, and to modify the output stride of the final layer by using dilated convolutions, such as proposed in [6], to be able to segment small objects at the expense of extra calculations. We use two different encoders to show that the gains of our approach are similar regardless of the architecture used, meaning that once a better feature extractor is designed, our method can be used on top of it.

Fig. 3. Instance labels and their corresponding 32-dimensional embeddings randomly projected to 3 dimensions to show them in RGB. Best viewed in color.

2) Semantic Segmentation Decoder: all decoders have the same architecture, but their last layer is passed through a different activation function. Furthermore, during training they are optimized with different losses. On top of the encoder we attach a module that upsamples the last layer's features and concatenates them to their matching skip resolution in the encoder (see Fig. 2). After the concatenation we use a $[1 \times 1]$ convolution that squashes the upsampled features with the skipped ones, and 2 layers of $[3 \times 3]$ convolutions to combine them and learn task-specific parameters. This is done iteratively until the output volume matches the size of the grid needed to perform the superpixel upsampling. For the semantic segmentation head, the output depth is the number of classes, and the activation function is a softmax $\hat{y}_c = \frac{e^{\text{logit}_c}}{\sum_c e^{\text{logit}_c}}$, where logit_c is the unbounded output in the slice corresponding to class c. This gives a pseudo-probability distribution per grid-element, which is optimized using a weighted cross-entropy loss:

$$L_{\text{semantic}} = -\sum_{c=1}^{C} w_c \; y_c \; \log\left(\hat{y}_c\right) \tag{1}$$

$$w_c = \frac{1}{\log\left(f_c + \epsilon\right)}; f_c = \frac{1}{P}\sum_{p=1}^{P} \begin{cases} 1 & \text{if } p = c \\ 0 & \text{if } p \neq c \end{cases} \tag{2}$$

where w_c which penalizes class c according to the inverse of its frequency in the ground truth, bounded by a parameter ϵ which is set to 1.02 in all our experiments.

3) Embedding Decoder: the objective of this branch is to provide a high-dimensional embedding (with 32 dimensions in our case) that has a low Euclidean distance to all other grid elements of the same instance, and a high Euclidean distance to all other instances. Fig. 3 shows this in action by randomly projecting all 32-dimensional embeddings into 3D space so that each embedding can be mapped to an RGB value (for illustrative reasons). The embedding branch is analog to the semantic segmentation branch, in terms of upsampling, concatenating, squashing, and adding extra

convolutional layers, but the main difference lies in the lack of an activation function. For this layer, instead of a softmax activation, we use the unbounded logits, which we call $\hat{e}$. Following [5] we define three hinged losses to achieve this purpose. In all equations, K is the number of instances, P_k is the number of grid elements in instance k, $\|\cdot\|$ is the L2-norm, and $[x]^+$ means the positive part of x, meaning $\max(0, x)$, which hinges the loss. The first loss is the attraction loss

$$L_{\text{attract}} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{P_k}\sum_{p=1}^{P_k}\left[\,\|\hat{e}_{k,c} - \hat{e}_{k,p}\| - \delta_{\text{a}}\right]^+ \quad (3)$$

which defines that all pixels of the same instance should have a low Euclidean distance to the embedding in its center. As in [5], this loss is hinged, which means that the embeddings that are already "close enough" to the center embedding do not receive a loss. This allows the embeddings to move around improving training and inference stability. This is modulated by the parameter δ_a, set to 0.1 for all our experiments. Note that in [5], the distance is calculated to the mean of the embedding due to the lack of the concept of object center. The second loss is the repelling loss:

$$L_{\text{repel}} = \frac{1}{K(K-1)}\sum_{\substack{k_A=1\\k_A\neq k_B}}^{K}\sum_{k_B=1}^{K}\left[\delta_{\text{r}} - \|\hat{e}_{k_A,c} - \hat{e}_{k_B,c}\|\,\right]^+ \quad (4)$$

This loss pushes the object centers from different instances away from each other in embedding space. Analogously to the attraction loss, this loss is hinged to allow the embeddings to move around when they are "far enough". This is modulated by δ_r, which is set to 1.0. The third loss $L_{\text{reg}} = \frac{1}{K}\sum_{k=1}^{K}\|\hat{e}_{k,c}\|$ penalizes the norm of all embeddings in the object centers to improve stability, making the embeddings stay close to the origin, and has no further meaning for the approach. We combine the three loss functions as a weighted sum and use it to backpropagate through the embedding decoder, as well as the encoder as: $L_{\text{embed}} = \alpha\, L_{\text{attract}} + \beta\, L_{\text{repel}} + \gamma\, L_{\text{reg}}$, with $\alpha = \beta = 1.0$ and $\gamma = 0.001$.

4) Instance Center Decoder: The last decoder head is the object center confidence head, which is analogous to the other two in terms of architecture design. This branch predicts, for each grid element, the confidence of it being the center of an object. In this decoder, the output volume is of depth 1, followed by a sigmoid activation of shape $\hat{y} = \sigma(\text{logit}) = 1/(1 + e^{\text{-logit}})$. This branch is optimized by a weighted cross-entropy loss between the output and the object center targets. However, because the number of grid elements is orders of magnitude larger than the average amount of objects in each image, the easy background elements overwhelm the loss. Therefore, we follow [16] and add an extra focal loss term modulated by γ. This makes the loss of easy background examples lower to prevent this overwhelming:

$$L_{\text{centers}} = \begin{cases} -\alpha\ (1-\hat{y})^{\gamma}\ \log(\hat{y}) & \text{if } y = 1 \\ -(1-\alpha)\ \hat{y}^{\gamma}\ \log(1-\hat{y}) & \text{if } y = 0 \end{cases} \quad (5)$$

In our experience, the object loss does not converge unless this term is added. Another important parameter for the confidence head is the initialization of the bias of the last layer before the sigmoid. Usually, models for binary classification are initialized to output positive or negative class with equal probability, and therefore, in the presence of imbalance as extreme as our task, the loss is dominated by the easy negatives causing instabilities. Following [16], we initialize the last bias of this decoder to $b = -\log((1-\pi)/\pi)$, where π is a prior calculated from the class imbalance. For all experiments, we use $\pi = 0.1$, $\alpha = 0.01$ and $\gamma = 2$, which are obtained as the parameter which gives the lowest cross-validation error.

B. Postprocessing

Once the CNN has predicted all three heads, we perform a fast post-processing step to obtain the final output. First, we mask the embeddings and object centers with the semantic segmentation grid from the first head, in order to separate the embeddings and centers of each class from each other and the background (see Fig. 2). Then we extract all grid elements that have a center confidence over 0.7 and extract the center of mass of all connected components in this binary mask. This step is necessary because the center of an object is not a perfectly defined concept, and therefore the CNN usually outputs blobs instead of single elements for each instance. We also perform an elimination of "duplicated" cluster centers, which are the centers that have embedding distances lower than δ_a from Eq. (3). These steps are analogous to the non-maximum suppression step in object detectors, where several anchors detect an object, and only one is kept by setting a threshold in the intersection over union.

Once we extract all the centers of objects, we mask their 32-dimensional embeddings as well as all the embeddings of the grid elements belonging to a certain class, and we calculate a similarity matrix between all elements in the class and all the centers. The next step calculates the maximum similarity object center for each element, and filters the ones that have a distance greater than δ_a, to eliminate elements that were incorrectly assigned by the semantic head. This procedure assigns a unique id to each instance in a class and groups all their elements, and is repeated for all classes. The final step in the post-processing is to upsample each grid element for both the clustered instances and the semantic mask into the original size output space, which is done using the one-to-many mapping exploiting the neighborhood color information from the superpixels (see Fig. 2). We explain how we obtain this mapping in the following section.

C. Superpixel Extraction for Locally Consistent Upsampling

Our approach upsamples the low resolution output grids of the CNN using an over-segmentation grid of the input in superpixels, improving over simple bilinear upsampling

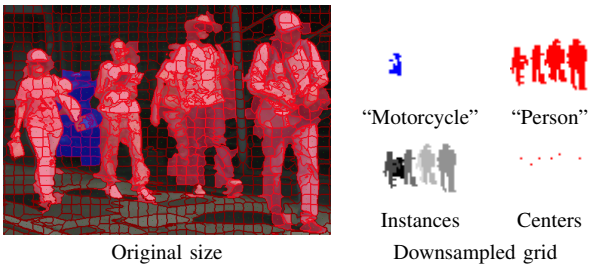

Fig. 4. Procedure to get from original sized labels to downsampled versions using the label at the center of mass of each superpixel.

without sacrificing runtime. These superpixels need to be small enough to capture the "connectivity" of real world objects, avoiding under-segmetation, but as big as possible in order to maximize the reduction of the size of the CNN output. This makes both, the inference and the clustering of the instances, faster. Thus, this size is a compromise which depends on the sizes of objects in the data, and is selected by evaluating the under-segmentation error in the training set of our data, the Cityscapes dataset [7].

For an image size $H \times W$ and a superpixel size k, the SLIC algorithm starts with a grid of size $H/k \times W/k$ of evenly distributed cluster centers c. The approach then iteratively (i) obtains the nearest neighbor cluster center using a distance $\delta_s = \|\mathcal{I}_{\text{Lab}}(i) - \mathcal{I}_{\text{Lab}}(c)\| + \alpha \|\mathcal{I}_{\text{xy}}(i) - \mathcal{I}_{\text{xy}}(c)\|$ for each pixel i, only looking in the clusters present in a $2K \times 2K$ neigborhood; and (ii) updates the cluster centers to become the mean of the newly associated cluster pixels. This is done until convergence, or until a limit number of iterations is met. Thanks to its definition as a selective nearest neighbor search followed by cluster center update, SLIC is highly parallelizable. The gSLICr implementation used in our approach performs the nearest neighbor search in CUDA using one thread per pixel, as well as the cluster update using a kernel to do the accumulation of the energy values, and a separate one for the reduction which returns the updated clusters. This allows for a segmentation which runs around 83 times faster than its CPU counterpart. Due to the rigid nature of the output of all CNNs, if we want to use the extracted superpixels to improve the boundary consistency of the output masks using a one-to-many mapping, we need all of the cluster centers in the over-segmentation to remain in a grid-like structure. Therefore, we perform one iteration of the algorithm in order to get the boundary information of each locally consistent area, but without updating the cluster centers. This yiels a grid in a structure that has a close resemblance with the regular grid with which the algorithm was initialized.

IV. Experimental Evaluation

The experiments are designed to show the efficiency and performance of our joint semantic and instance segmentation approach, and to support the claims that our method is capable of performing both tasks in parallel, accurately, and faster than previous approaches. We implemented the whole approach presented in this paper relying on Pytorch.

A. Training Data

For our experiments, we use the Cityscapes dataset [7], which contains 5,000 annotated images with fine annotations of 30 semantic classes, 8 of which contain instance labels, plus 20000 images containing coarse annotations. For our experiments, we use the 8 classes containing fine pixel-wise instance information: "person", "rider", "car", "truck", "bus", "train", "motorcycle", and "bicycle", and we treat the remaining pixels as background, both for the semantic as well as the instance task (see Fig. 5 for examples of the images in the dataset). To train our architecture, we choose a downsampling rate based on two criteria: maximizing the superpixel size to decrease the output stride of the decoders, making the clustering of the instance elements faster, and at the same time, minimizing the under-segmentation error. For all experiments, we use an image size of 1024×512 and the best performing superpixel size in our experiments was 4, which allows us to reduce the number of pixels in the post-processing by 16, as well as reduce the inference time of the CNN. We train our networks downsampling the targets using the superpixels extracted from the inputs. This is, if an image and its corresponding semantic and instance label are of size $H \times W$, and the superpixels are of size k, the output grids and the used targets for each loss are of size $H/k \times W/k$ and each target grid element is the label at the center of mass of its corresponding superpixel (see Fig. 4). We train the whole architecture end-to-end starting with the encoder pretrained on ImageNet, as stated in Sec. III-A.

B. Performance

This experiment is designed to show that our algorithm can efficiently segment and classify individual objects from camera images, without sacrificing accuracy.

Tab. I shows the results of training in the 2,975 training set images with fine, pixel-wise annotations, and validating the results in 500 held out images from different cities. The first row shows the results of training an architecture with a strong, state of the art backbone (DN53) and inferring only the semantic and embedding branch, without the embedding centers or superpixel upsampling, and applying an all-to-all clustering of the instance pixels afterwards. The subsequent rows show our approach inferring the cluster centers with different backbones, and decoder output strides. The results show two interesting effects. First, even though the baseline with no superpixel upsampling or center inferrence performs slightly better than our best performing architecture (row 0 vs. 1), this costs almost 3 times more to run, due to the expensive post-processing in the absence of the inferred cluster centers. Second, when using decoder OS 4, which means upsampling with superpixels of size $k = 4$, the models perform similarly to their non-upsampled counterparts, but run 2 times faster in the case of the Darknet based model, and almost 3 times faster for the Mobilenets based one.

Tab. II and Tab. III show a comparison to other state-of-the-art methods for semantic segmentation and instance segmentation, respectively. Tab. II shows that our algorithm performs the semantic segmentation task on par with other

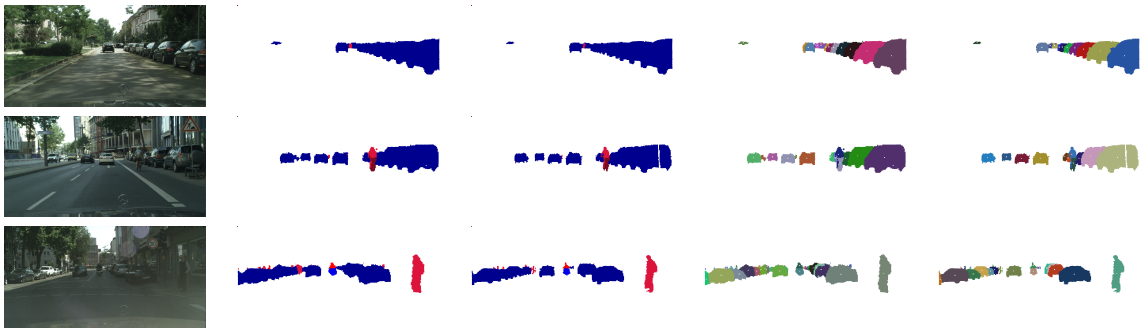

Fig. 5. Examples of results on the validation set for our MobilenetsV2 backend using a decoder output grid downsampled 16 times (decoder OS 4).

TABLE I

PERFORMANCE ON VALIDATION SET BY ENCODER, INPUT SIZE, AND OUTPUT STRIDES

Encoder	Parameters			Instance		Semantic	Runtime (avg.)			
	Input	Enc. OS	Dec. OS	mAP@50%	mAP	IoU	Spix	CNN	Cluster	FPS
Baseline using Darknet 53	1024 x 512	16	1	46.9%	24.3%	65.4%	-	138 ms	240 ms	2
Ours using Darknet 53	1024 x 512	16	1	46.7%	22.3%	65.4%	-	141 ms	36 ms	5.5
			4	46.5%	21.9%	63.7%	4 ms	73 ms	9 ms	12
Ours using Mobilenets V2	1024 x 512	16	1	46.2%	21.9%	62.3%	-	48 ms	36 ms	11
			4	45.2%	21.1%	60.4%	4 ms	19 ms	9 ms	31

state-of-the-art, real-time methods, but with the burden of having to infer the instances as well. Tab. III shows that our approach performs better than other state of the art methods, but slightly under-performs the best benchmark submissions such as MaskRCNN [12] and PANet [17], which are roughly 15 times slower to run.

TABLE II

COMPARISON WITH OTHER STATE-OF-THE-ART REAL-TIME SEMANTIC SEGMENTATION METHODS.

IoU	Segnet*[2]	ERFNet*[28]	ENet*[23]	Ours MNv2	Ours DN53
Car	89.2%	93.4%	91.0%	88.6%	94.1%
Bus	43.1%	60.8%	49.3%	77.3%	73.2%
Truck	38.1%	52.2%	39.3%	57.4%	62.1%
Motorc.	35.7%	49.8%	41.6%	38.9%	45.8%
Train	44.1%	53.7%	50.5%	61.9%	62.4%
Bicyc.	51.8%	64.2%	59.8%	48.6%	53.9%
Person	62.7%	78.5%	71.3%	60.2%	61.3%
Rider	42.8%	59.7%	49.6%	50.6%	57.3%
Mean	50.9%	64.0%	56.5%	60.4%	63.7%
FPS	16	50	76	31	12

Methods with * perform the semantic task exclusively, not predicting instances.

TABLE III

COMPARISON WITH OTHER STATE-OF-THE-ART INSTANCE SEGMENTATION METHODS.

Model	AP	AP@50%	FPS
DeepWatershed[3]	19.4%	35.3%	-
FSUfAD[22]	21.0%	38.6%	21
Mask-RCNN [12]	26.2%	49.9%	2
PANet[17]	31.8%	57.1%	2
Mask-RCNN* [12]	32.0%	58.1%	2
PANet*[17]	36.4%	63.1%	2
Ours MNv2	21.1%	45.2%	31
Ours DN53	21.9%	46.5%	12

Methods with * were pretrained with COCO [15] dataset.

V. CONCLUSION

In this paper, we presented novel approach for joint semantic and instance segmentation in an efficient manner. Our main contribution is a CNN architecture based on one-shot object detection and metric learning that can perform the semantic and instance segmentation tasks simultaneously in real-time. We achieve this by performing a fast GPU-based superpixel over-segmentation, which allows us to exploit local connectivity of neighboring pixels and reduce the complexity of our algorithm, while still obtaining high resolution masks. Our one-shot object detection based pipeline jointly predicts the confidence of each superpixel being the center of an object, its semantic class, and a high-dimensional embedding which allows us to assign each individual occupied superpixel to an instance center. This allows us to achieve three important algorithmic qualities. Firstly, given the over-segmentation grid and the regression of the instance centers as an object confidences, we avoid the selection of the number of cluster centers for the k-means clustering of the embeddings, and instead we can just count the regressed instance centers over a desired confidence level. Secondly, by exploiting the neighborhood information we both reduce the number of pixels to cluster, allowing us to use bigger images, and avoid stranded pixels at test time. Finally, the one-shot architecture combined with the neighborhood summarization makes the extraction of the instances faster when compared to two-stage as well as other metric learning based methods.

ACKNOWLEDGMENTS

We thank NVIDIA for providing a Quadro P6000 GPU which was partially used to train the models in this paper, and Philipp Lottes and Jens Behley for fruitful discussions.

REFERENCES

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Ssstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 2012.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 2017.

[3] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] N. Blodow, L. C. Goron, Z. C. Marton, D. Pangercic, T. Rühr, M. Tenorth, and M. Beetz. Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4263–4270, 2011.

[5] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation with a discriminative loss function. In *Deep Learning for Robotic Vision workshop, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint*, 2017.

[7] M. Cordts, S. Mohamed Omran, 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 Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, June 2009.

[9] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C.C. Lerma. SegMatch: Segment Based Place Recognition in 3D Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[10] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H.O. Song, S. Guadarrama, and K.P. Murphy. Semantic Instance Segmentation via Deep Metric Learning. *arXiv preprint*, 2017.

[11] S. Garg, N. Snderhauf, and M.J. Milford. Don't look back: Robustifying place categorization for viewpoint and condition-invariant place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint*, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] R. Khanna, M. Möller, J. Pfeifer, F. Liebisch, A. Walter, and R. Siegwart. Beyond point clouds - 3d mapping and field parameter measurements using uavs. In *Proc. of the IEEE Conf. on Emerging Technologies Factory Automation (ETFA)*, pages 1–4, 2015.

[15] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 740–755, 2014.

[16] T.Y. Lin, P. Goyal, R.B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint*, abs/1708.02002, 2017.

[17] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] P. Lottes, M. Höferlin, S. Sander, M. Müter, P. Schulze-Lammers, and C. Stachniss. An Effective Classification System for Separating Sugar Beets and Weeds for Precision Farming Applications. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[19] J. McCormac, R. Clark, M. Bloesch, A.J. Davison, and S. Leutenegger. Fusion++: Volumetric Object-Level SLAM. *arXiv preprint*, 2018.

[20] A. Milioto, P. Lottes, and C. Stachniss. Real-time blob-wise sugar beets vs weeds classification for monitoring fields using convolutional neural networks. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2017.

[21] A. Milioto, P. Lottes, and C. Stachniss. Real-time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[22] D. Neven, B.D. Brabandere, S. Georgoulis, M. Proesmans, and L.V. Gool. Fast Scene Understanding for Autonomous Driving. *arXiv preprint*, 2017.

[23] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: Deep neural network architecture for real-time semantic segmentation. *arXiv preprint*, 1606.02147, 2016.

[24] C. Payer, D. tern, T. Neff, H. Bischof, and M. Urschler. Instance Segmentation and Tracking with Cosine Embeddings and Recurrent Hourglass Networks. *arXiv preprint*, 2018.

[25] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint*, 2018.

[26] C.Y. Ren, V.A. Prisacariu, and I.D. Reid. gSLICr: SLIC superpixels at over 250Hz. *arXiv preprint*, 2015.

[27] M. Ren and R.S. Zemel. End-to-end instance segmentation and counting with recurrent attention. *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[28] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Trans. on Intelligent Transportation Systems (ITS)*, 19(1):263–272, 2018.

[29] B. Romera-Paredes and P.H.S. Torr. Recurrent instance segmentation. *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, abs/1511.08250, 2016.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arxiv*, 2018.

[31] M. Schwarz, A. Milan, A.S. Periyasamy, and S. Behnke. Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter. *Intl. Journal of Robotics Research (IJRR)*, 2017.

[32] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P.I. Corke, G. Wyeth, B. Upcroft, and M. Milford. Place categorization and semantic mapping on a mobile robot. *arXiv preprint*, abs/1507.02428, 2015.