Instance Segmentation with Unsupervised Adaptation to Different Domains for Autonomous Vehicles

Manuel Alejandro Diaz-Zapata, Özgür Erkent, Christian Laugier

Instance Segmentation with Unsupervised Adaptation to Different Domains for Autonomous Vehicles

Manuel Diaz-Zapata[1], Özgür Erkent[1], Christian Laugier[1]

***Abstract*— Detection of the objects around a vehicle is important for a safe and successful navigation of an autonomous vehicle. Instance segmentation provides a fine and accurate classification of the objects such as cars, trucks, pedestrians, etc. In this study, we propose a fast and accurate approach which can detect and segment the object instances which can be adapted to new conditions without requiring the labels from the new condition. Furthermore, the performance of the instance segmentation does not degrade in detection of the objects in the original condition after it adapts to the new condition. To our knowledge, currently there are not other methods which perform unsupervised domain adaptation for the task of instance segmentation using non-synthetic datasets. We evaluate the adaptation capability of our method on two datasets. Firstly, we test its capacity of adapting to a new domain; secondly, we test its ability to adapt to new weather conditions. The results show that it can adapt to new conditions with an improved accuracy while preserving the accuracy of the original condition.**

I. Introduction

An autonomous vehicle needs to detect the number and location of objects around it to perceive and comprehend its surroundings. Although several object detection algorithms propose to find the objects, this results in bounding boxes which is a coarse localization of the objects ([1], [2]). In this study, we deal with instance segmentation and its unsupervised adaptation to new domains in this study.

Instance segmentation can be defined as the detection of the pixels in an image belonging to an instance such as car, truck, pedestrian, rider, etc. It is considered to be harder than object detection or semantic segmentation since all the object instances must be found in an image and a bounding box is not sufficient [3]. This is mainly due to the fact that in instance segmentation, you need to count the number of occurrences of an instance in an image, contrary to semantic segmentation where you don't discriminate between different instances of the same class, i.e. you cannot discriminate between two instances of a vehicle. Due to this difficulty, instance segmentation usually provides more detailed information with respect to semantic segmentation and object detection with the cost of an increased computation time. Since we are considering autonomous vehicles, we propose an approach which is fast and achieves a decent accuracy to be used on an autonomous vehicle.

An important problem with modern approaches for instance segmentation methods which depend on deep neural networks (DNNs) is their requirement for tremendous amount of data for training. Utilization of the same model in different conditions usually results in reduced performance. Although the performance can be increased by labeling the images from the new condition, this will increase the cost of training since labeling at pixel level is a high-cost task [5] which can take up to several hours for a single image.

To overcome this problem, we consider the unsupervised domain adaptation (UDA) which adapts the previously obtained DNN model into the new condition without requiring labels ([6], [7]). We achieve this adaptation by making the features of both of the domains similar to each other in the

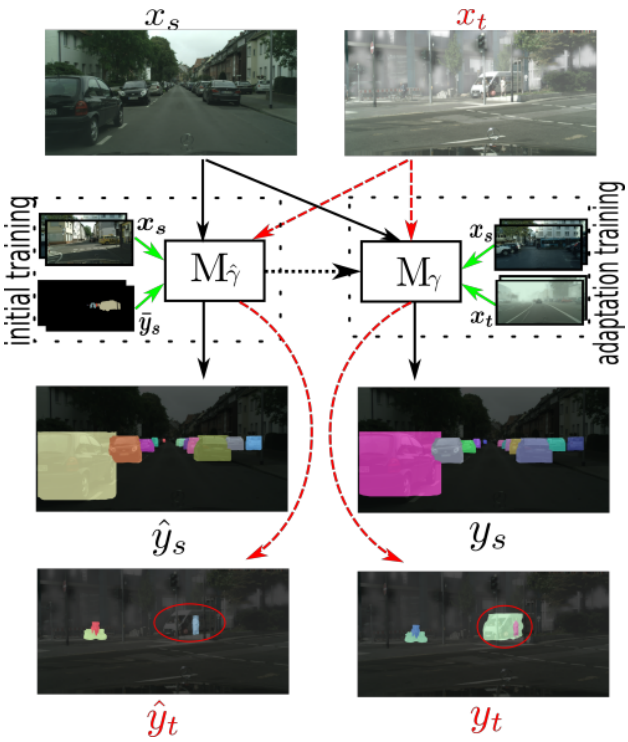

Fig. 1: A sample overview of the instance segmentation adaptation. The images are from the Cityscapes Dataset [4] and its foggy variant [4]. A model based on YOLOv3 is used [2]. $M_{\hat{\gamma}}$ is the model trained for the source domain (original condition), M_{γ} is the adapted model based on only RGB images. x_s: source domain, x_t: target domain, $\bar{y}_s$: ground truth label that was used only for original model training, $\hat{y}_s$, $\hat{y}_t$: output of the $M_{\hat{\gamma}}$ on source and target domain, y_s, y_t: output of M_{γ} on source and target domain. The red circle shows the improvement (minivan detected after adaptation) in target domain instance segmentation.

This work was supported by EU project CPS4EU.

[1] Authors are with Univ. Grenoble Alpes, INRIA, Chroma Team, France INRIA, Rhône-Alpes, France. Correspondence: name.surname@inria.fr

initial layers of the network (see Fig. 1). In this context, source domain refers to the old condition where we train our model and target domain refers to the new condition without labels.

The contributions of this work can be listed as follows:

- A new instance segmentation method based on YOLOv3 [2] which is fast and achieves a high accuracy
- An unsupervised domain adaptation method for instance segmentation which does not require the labels neither from source nor target domain
- The adapted instance segmentation model can still achieve a high accuracy in the source domain.

We evaluate our approach on two datasets for instance segmentation and show that the proposed approach can achieve results similar or even better than supervised methods.

In Section. II, we briefly summarize the related literature; in Section. III, we explain the new instance segmentation method; in Section. IV, the instance segmentation domain adaptation method is explained and in Section. V, we provide the results of our evaluation on two datasets. Finally, we summarize the findings of our work.

II. Related Work

First, we make a brief summary of instance segmentation methods and then we provide the previous studies on unsupervised domain adaptation in general with a specific focus on object detection and segmentation.

A. Instance segmentation

Currently, instance segmentation methods follow one of two variants: pixel-based methods and polygon-based methods. In this work, a pixel-based algorithm is proposed.

1) Pixel-based methods: Pixel-based methods gained a high popularity in recent years due to their success. Here, instances are described by masks that indicate which pixels belong to them. The creation of these masks is usually aided by region proposals from a parallel detector such as Faster R-CNN [1] in Mask R-CNN [8] and a cascade variant of Mask R-CNN [9], where an FCN [10] is used to perform the foreground segmentation on the bounding box predictions in order to find which pixels belong to the object. In YOLACT [11], instead of focusing its attention to specific parts of the image using the bounding boxes from a detector stage, prototypes for the entire image are found and linearly combined using a set of predicted coefficients.

Other instance segmentation methods such as AdaptIS [12] and Deeperlab [13], follow a keypoint-based approach. In AdaptIS, the network adapts to an input point using adaptive instance normalization layers [14], creating masks in an iterative process for each of the objects in the scene. Deeperlab models the relationships between predicted keypoints and the pixels of the instance using a keypoint heatmap and long, medium and short-range offset maps.

Methods like Recurrent Instance Segmentation [15], performs sequential segmentation of the objects in an image and keeps track of which pixels have been assigned with a Convolutional LSTM module.

2) Polygon-based methods: Instead of predicting the pixels that belong to an instance, some methods predict instead the polygon that best fits the instance's boundary, which can help to reduce computation time. ESE-Seg [16] performs a Chebyshev polynomial fitting by adding a parallel decoder to a YOLOv3 [2] network. In the case of the deep snake [17] approach, the network takes a contour as input and outputs vertex-wise offsets to deform the contour in order to match the boundary of the detected instance.

We implement a pixel-based instance segmentation method due to their high accuracy; however, since their speed is generally low for the autonomous vehicles, we propose a new approach which can adapt to new conditions.

B. Unsupervised Domain Adaptation

As aforementioned, deep learning methods require tremendous number of labeled images. However, it is not possible to label images for each new condition, such as a new city, a new vehicle or a new weather condition. Therefore, a number of studies have proposed to use unsupervised domain adaptation where the labeled images from the new domain are not available.

For example, a group of studies adapt to a new condition by using a small number of samples from the new domain and incrementally update the available network. However, this needs an incremental change in the conditions. For example Dai *et al*. [18] use different times of twilight images from lighter to darker hours of twilight for adaptation to darkness. Wulfmeier *et al*. [19] also use an incremental approach for adaptation to darkness. Since new domains do not always have the incremental changes, we don't follow such an approach.

Another possibility is to use the invariance of the sensors to the weather conditions such as the invariance property of laser range sensors in different weather conditions and dark. Kim *et al*. [20] use Lidars in different weathers. Chen *et al*. [21] propose to use the depth information to adapt from synthetic domain to real-world domain. Erkent *et al*. [22] also use the the semantic 2D maps to obtain an semantic grid of the scene. However, it is not always possible to find sensors which will be invariant to new weather conditions, therefore they need to be trained for each condition.

Another approach is to make the output of both domains similar to each other. For example, Vu *et al*. [23] use GAN training [24] on the entropy of the output which is used to make these distributions similar to each other. However, if you measure the similarity on the output only, you need to convert these outputs into a high-dimensional space which will increase computation time.

Another line of work deals with the similarity of the distribution of the features since they already have a high-dimensional data. Long *et al*. [6] is one of the early works which consider to update the neural network to obtain similar features for domain adaptation. GANs are also used to make the features similar to each other which was initially proposed by Ganin *et al*. [25]. Due to its success, it has been implemented by several studies including [26], [27], [28] for

semantic segmentation. Zhang *et al*. [29] use it for instance segmentation from synthetic to real domain . Adaptation approaches do not always use GANs, for example Peng *et al*. [30] implement Wasserstein GANs (W-GANs) [31], Erkent *et al*. [7] use W-GANs and MMDs for adaptation of semantic segmentation.

Other than output and feature similarity, some studies try to make the inputs similar to each other before they are provided to the classification network. Cycada [32] trains in Synthetic data and then transforms the real images into a synthetic style by using GANs. [33] and [34] also use a similar approach for semantically segmenting the scenes. It should be noted that this requires additional computation to pre-process the images.

Our approach, which adapts the instance segmentation network into new conditions is different from the aforementioned approaches in a few aspects. First, we propose a method which can preserve a similar accuracy for source domain while increasing the accuracy for the target domain for instance segmentation thanks to self-supervision loss. Secondly, we do not require the instance labels for neither source nor target domain for adaptation. In instance segmentation, converting the instance labels from one dataset to another can be a time-consuming task depending on the labeling conventions, or even impossible depending of the availability of the source domain labels.

III. Instance Segmentation Network

We use a model based on YOLOv3 [2] for instance segmentation. Here, a segmentation head is added to the Darknet-53 backbone in parallel to the YOLO detection head. This segmentation head uses the Pyramid Pooling Module (PPM) proposed in PSPNet [35] to produce enriched features.

First, the network takes three feature maps from different depths in the backbone to find multi-scale information. Upsampling is performed on the two feature maps with smaller spatial size, in order to match the shallowest feature map for concatenation. A 1×1 convolution is also applied to the shallowest map in order to increase its channel depth.

Once the feature maps are concatenated, average pooling is performed with bin sizes of $[1, 2, 3, 6]$ together with a 1×1 convolution to reduce depth size by four, batch normalization and ReLU activation. The resulting four feature maps are appended to the original feature map, resulting in the set of features that will be used for instance segmentation. Two separate convolutions are applied to the resulting feature map, one to perform semantic segmentation and another one to decrease feature depth. Semantic segmentation is necessary to separate the foreground from the background for the instance segmentation.

With the semantic feature maps processed, the bounding boxes from YOLO are used to extract the detections to do foreground segmentation. First, non-maximum suppression is performed on the bounding boxes proposed by the detector head. Then, since YOLO's predictions are relative to the image size, the bounding boxes are scaled to the size of the feature map to crop the detections for instance segmentation. This processing of the features can be seen on Fig. 2.

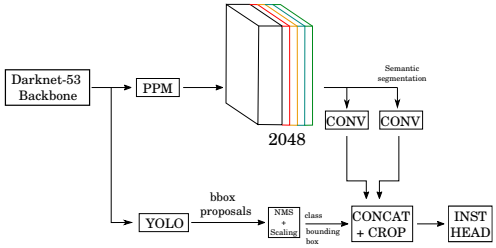

Fig. 2: Proposed instance segmentation network.

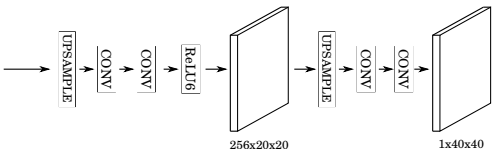

Fig. 3: Proposed instance segmentation head.

Finally, foreground segmentation is performed in the same fashion as in [8], where an FCN architecture is used to find the mask (Fig. 3). The cropped features are first bilinearly upsampled to have a spatial size of 20×20, then two 3×3 convolutions with stride of 1 are applied followed by a ReLU non-linearity. The resulting feature map is upsampled once more to a size of 40×40, followed by a 3×3 convolution and a 1×1 convolution, both with stride of 1, to find the instance mask.

IV. Domain Adaptation for Instance Segmentation

We will make a brief reminder of the unsupervised domain adaptation (UDA) and explain how it is used for instance segmentation. First, we define a couple of terms. "*Source domain*": $\mathcal{S} = \{(\mathbf{x}_s^i, \bar{y}_s^i)\}_{i=1}^{n_s}$, "*target domain*": $\mathcal{X}_\mathrm{t} = \{(\mathbf{x}_t^j)\}_{j=1}^{n_t}$, "*source data*" (no labels): $\mathcal{X}_\mathrm{s} = \{(\mathbf{x}_s^i)\}_{i=1}^{n_s}$, "*target data*": $\mathcal{X}_\mathrm{t}$ with n_s and n_t the number of samples for source and target domain respectively. A deep neural network is trained on "*source domain*" $\mathrm{M}_{\hat{\gamma}}$ to detect the instance of each pixel $\hat{y}_s^i$. The aim of $\mathrm{M}_{\hat{\gamma}}$ is to make a prediction such that the cost $c_s\left(\mathrm{M}_{\hat{\gamma}}\right) = \Pr_{(\mathbf{x},\bar{y})\sim p}\left[\mathrm{M}_{\hat{\gamma}}(\mathrm{x}) \neq \bar{y}\right]$ is minimized.

In UDA, since we don't have access to labels of the target domain, we try to approximate the distribution of the target domain to the source domain. For this, we investigate the probability distributions of the source and target domain $\Pr_p$ and $\Pr_q$. The model M_γ is updated such that the features of both domains become similar to each other while the c_s is simultaneously minimized. An overview of the proposed adaptation method is given in Fig. 4. $\mathrm{M}_{\hat{\gamma}}$ is the initial model which is trained on source domain with its provided labels. This model can also be a pre-trained network. We separate this model into two parts: $\mathrm{E}_{\hat{\theta}}$ and F_α. γ, θ, $\hat{\gamma}$, $\hat{\theta}$ and α are the hyperparameters of the neural networks. $\mathrm{M}_{\hat{\gamma}}$ is replicated with its parameters as M_γ. While the parameters of $\mathrm{E}_{\hat{\theta}}$ and $\mathrm{F}_{\hat{\alpha}}$ are fixed, the parameters of E_θ are optimized so that the output of E_θ for target domain

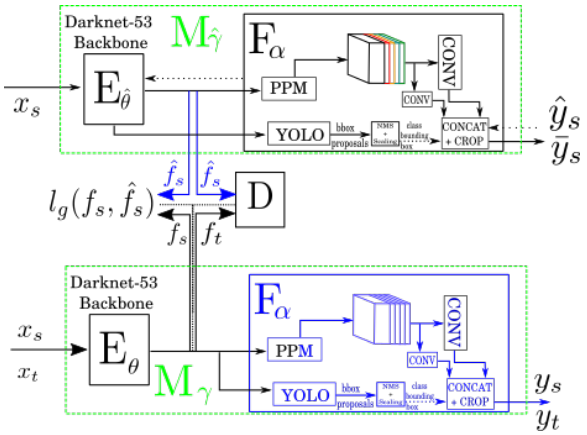

Fig. 4: Adaptation schema overview. Green dotted boxes are the simplified instance segmentation models. Dotted black lines use the back-propagation; while, the blue lines are the processes without back-propagation.

data become similar in distribution to the source domain data. Therefore, the target cost optimization can be rewritten as $c_t\left(\mathrm{E}_\theta(\mathbf{x}_t)=\mathbf{f}_t\right) = \Pr_{(\mathbf{f},y)\sim q}\left[\mathrm{F}_\alpha(\mathbf{f})\neq y\right]$ and hence the parameters to be optimized reduce from $||\gamma||$ to $||\theta||$. This results in reduced training time since we need to optimize the parameters of a smaller network.

It should be noted that the capacity of E must be sufficient to produce similar features for both domains while providing features for a high quality classification. The optimization of θ can be given as:

$$\inf_{\theta\in\Theta} \mathrm{D}(\Pr_{(\hat{\mathbf{f}}_s)\sim p}, \Pr_{(\mathrm{E}_\theta(\mathbf{x}_t))\sim q}) + \lambda l_g(\hat{\mathbf{f}}_s, \mathrm{E}_\theta(\mathbf{x}_s)) \quad (1)$$

where $\theta\in\Theta$ and $\mathbf{f}_t^i = \mathrm{E}_\theta(\mathbf{x}_t^i)$ are target domain features. D is the network that gives the distribution between source and target domain features. $\lambda>0$ is used as a regularization hyperparameter; $l_g(\hat{\mathbf{f}}_s, \mathrm{E}_\theta(\mathbf{x}_s)) = ||\hat{\mathbf{f}}_s - \mathrm{E}_\theta(\mathbf{x}_s)||_2^2$ is the self-supervision loss.

We measure the distance $\mathrm{D}(\Pr_{(\hat{\mathbf{f}}_t)\sim p}, \Pr_{(\mathrm{E}_\theta(\mathbf{x}_t))\sim q})$ between the source and target domain feature distributions by using the Wasserstein Distance.

W-GAN: $m-$th order Wasserstein distance is [31]:

$$W_m(\Pr_{(\hat{\mathbf{f}})\sim p}, \Pr_{((\mathbf{f}))\sim q}) = \inf_{\Pr_{\hat{\mathbf{f}}_s\sim p,(\mathbf{f}_t)\sim q}} \mathbb{E}\left[||\hat{\mathbf{f}}_s - \mathbf{f}_t||^m\right] \quad (2)$$

over all joint distributions. This optimal transport problem over joint probabilities has a dual formulation (Kantorovich-Rubinstein) [36]. It can be rewritten for $m=1$ as follows:

$$\inf_{\theta\in\Theta}\inf_{\phi\in\Phi} \mathbb{E}\left[\mathrm{D}_\phi(\hat{\mathbf{f}}_s)\right] - \mathbb{E}\left[\mathrm{D}_\phi(\mathrm{E}_\theta(\mathbf{x}_t))\right] + \lambda l_g(\hat{\mathbf{f}}_s, \mathrm{E}_\theta(\mathbf{x}_s)) \quad (3)$$

where $\phi\in\Phi$ represents the weights of the discriminator D. A solution to find the parameters of the discriminator has been proposed by Arjovsky *et al.* [31]. Furthermore, gradient penalty variant solves the problems related to convergence

Algorithm 1: Self-Supervised Domain Adaptation

Require: $\mathcal{X}_s$, $\mathcal{X}_t$, $\mathrm{E}_{\hat{\theta}}$, E_θ, D_ϕ, $\lambda>0$, n: Mini-batch size, n_{crit}, $\lambda_p>0$: Penalty gradient
Initialize $\theta=\hat{\theta}$
Initialize the parameters ϕ of D_ϕ
while *(θ) not converged* **do**
  Sample $\{(\mathbf{x}_s^i)\}_{i=1}^n \in \mathcal{X}_s$, $\{(\mathbf{x}_t^i)\}_{i=1}^n \in \mathcal{X}_t$
  Compute $\hat{\mathbf{f}}_s^i = \mathrm{E}_{\hat{\theta}}(\mathbf{x}_s^i)$, $\mathbf{f}_t^i = \mathrm{E}_\theta(\mathbf{x}_t^i)$, $\mathbf{f}_s^i = \mathrm{E}_\theta(\mathbf{x}_s^i)$;
  for $k=1,\cdots,n_{crit}$ **do**
    Sample a random number $\epsilon\sim U[0,1]$;
    $\widetilde{\mathbf{f}}^i = \epsilon\hat{\mathbf{f}}_s^i + (1-\epsilon)\mathbf{f}_t^i$;
    Update D_ϕ by ascending

$$\frac{1}{n}\sum_{i=1}^n \mathrm{D}(\hat{\mathbf{f}}_s^i) - \mathrm{D}(\mathbf{f}_t^i) - \lambda_p(||\nabla_{\widetilde{\mathbf{f}}}\mathrm{D}(\widetilde{\mathbf{f}}^i)||_2 - 1)^2$$

    Update E_θ by descending

$$\frac{1}{n}\sum_{i=1}^n \mathrm{D}(\mathbf{f}_t^i) + \lambda l_g(\mathbf{f}_s^i, \hat{\mathbf{f}}_s^i)$$

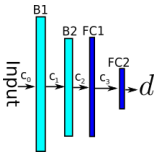

Fig. 5: Discriminator D. B1 and B2: 2D convolutional + instance normalization layers [40]. FC1 and FC2: fully connected layers. $c_0=512$, $c_1=c_2=64$ and $c_3=262144$ are number of channels and FC2 outputs a scalar value.

which we use in this study [37]. For a detailed understanding of the algorithm, please refer to Algorithm. 1.

More details and variants of the algorithm can be found in [7] with further experiments for the semantic segmentation problem.

V. Experiments

We evaluate the adaptation capability of our instance segmentation method on two different domains: Semantic KITTI [38] and Foggy Cityscapes Dataset (CS) [4]. We use the weights of the YOLOv3 network [2] which is trained on the COCO Dataset [39]. The weights of the instance segmentation part are obtained by using the Cityscapes Dataset; however, since we freeze the YOLO object detection network, the object detection is not improved with this training and certain classes such as *rider* is not detected.

We use the source domain as Cityscapes in both of the datasets. For unsupervised domain adaptation the hyper parameters are selected to be $n_{crit}=5$, $\lambda=10$, $\lambda_g=20$ and Adam optimizer with learning rate $\alpha=1\times10^{-4}$ and $\beta_1=0.7$ and $\beta_2=0.9$. The time requirement of inference for images with size of 640x640 pixels is 66 ms while for 1024x1024 pixels, it is 165 ms. In the experiments, we use 1024x1024 pixel sized images. We use a shallow discriminator D as shown in Fig. 5.

For evaluation, we use AP and AP 50 % measures

where AP is the average precision over the region level for each class. For AP, 10 different overlaps are used between $[0.5, 0.95]$ in steps 0.05. Overlap is computed for each instance at region level. AP 50 $\%$ is also provided which gives the average precision with 50 $\%$ overlap.

Semantic KITTI [38] is a variant of KITTI dataset where 200 images are selected and labeled in detail with same labels of Cityscapes. Although both datasets are obtained from the cameras of moving vehicles in traffic, the properties of the cameras and labeling strategies make these two datasets significantly different from each other and the datasets that are trained and achieve a high accuracy of instance segmentation accuracy cannot achieve the same high accuracy on KITTI instance segmentation as demonstrated on the benchmark [1]. We use our network which is trained for instance segmentation on Cityscapes, and adapt it to this dataset by using 200 images of the KITTI dataset without using any of the labels. In the current benchmark, no unsupervised method has been reported until now; therefore, this is the first time UDA is performed on Semantic Instance Segmentation KITTI. The image sizes of Cityscapes and KITTI are different. We test the images with and without resizing the KITTI images to be in similar size of CS and find out that resizing improves the accuracy, therefore we use the resized images for the adaptation as shown in Table. I.

Results: As it can be seen, the average performance is highest when the UDA is applied on the method; however, this is not the case for all the classes. For example, the accuracy of the truck and bus detection reduces. This is probably due to the small number of occurrences of these classes. The trains are also recognized worse with respect to non-resized cases probably due to the visibility of the trams. There are only a few frames where these rare classes are visible. We don't report the rider class since we use the pretrained Coco dataset which has no rider class. In overall, the results are comparable with or better than some the state-of-art results in the KITTI benchmark although no supervision is used from the KITTI dataset. Furthermore, the accuracy of the source domain, CS dataset, drops from AP $\%50 = 25.7\%$ to 24.8% which is less than the increase in KITTI.

The visualization of four samples are shown in Fig. 6. The ground truth is shown on the left column, the non-adapted resized samples in the middle and the adapted ones are on the right column. The important differences between them are shown with a red circle. On the top row, two vehicles are detected as single before adaptation. On the second row, a person is hallucinated before the adaptation. On the third column, a human is detected after adaptation correctly. In the last row, a vehicle which is not detected at a distance is detected after adaptation. It should be noted that although some of the results are corrected after adaptation, it is not sufficient to correct all. The accuracy of the initial model is important to achieve a better adaptation.

[1]http://www.cvlibs.net/datasets/kitti/eval_instance_seg.php?benchmark=instanceSeg2015

Foggy Cityscapes [4], [5] is a collection of images in which the fog is overlaid to the original real images of the Cityscapes dataset. Therefore, it is a realistic synthetic dataset. The only additional noise factor is the fog, the camera parameters are same in both; therefore we don't resize the images. Three different variances of fog are provided in the CS Foggy dataset. Fog-02 refers to heavy fog while Fog-01 refers to lighter and Fog-005 refers to the lightest fog conditions. An example of three conditions is given in Fig. 7 with the original condition.

For source domain, we use 2975 training images of the CS, while for the target domain, 1500 validation images of foggy condition are used. We use a mixture of all foggy weather conditions since in real life, it is not possible to estimate the amount of fog easily. Again, no labeled images are used for adaptation training and the same parameters with KITTI UDA are used.

Results The results improve significantly for Fog-02 as it can be seen in Table. II. The accuracy of Fog-005 also improves slightly. However, the accuracy of Fog-01 does not improve. The slight or no change in the performance for Fog-01 and Fog-005 conditions can be attributed to the high similarity of source and target domains for instance segmentation. Furthermore, the performance on the original CS dataset does not degrade significantly (from 25.7 % to 24.6 %). To understand the effect of adaptation in detail, we further provide data for Fog-02 condition in Table. III. As it can be seen, nearly all classes improve in performance; only, bus class which has a few samples reduce slightly in performance. Although the detections of cars improve slightly, it is necessary to note that the car class has many samples; therefore, the increase in car class shows a significant increase in the overall dataset.

To visualize the effects of the adaptation, we show some of the results in Fig. 8. On the top row, the tram is recognized after adaptation, which is important in the scene. On the second row, a human is detected on a bike after detection. Since we don't have the rider class in training, this is not recognized as rider. On the third row, again the tram is recognized after adaptation. Finally, on the last row, a minivan is recognized after adaptation which is an important object in the scene.

VI. Conclusion

Our proposal depends on two main components: a fast instance segmentation network and an unsupervised domain adaptation approach. As it can be seen, a DNN, which uses Darknet as a backbone can be adapted to new conditions for instance segmentation. The performance increases significantly for the cases where the difference between the source and the target domain is significant such as thick fog condition, or a completely new dataset. For the KITTI dataset, our adaptation outperforms even some of the supervised methods in the KITTI benchmark. As previously mentioned, currently there are not other methods apart from this one which perform unsupervised domain adaptation between two real datasets for the task of instance segmentation. In the

TABLE I: Results with CRF on the KITTI dataset with different architecture types.

	No Resize		Resized		Resized, Adapted	
(%)	**AP**	**AP 50%**	**AP**	**AP 50%**	**AP**	**AP 50%**
person:	1.00	6.70	3.60	16.10	**5.60**	**26.10**
car:	19.40	43.20	16.00	38.30	**27.30**	**58.10**
truck:	3.00	4.70	**12.80**	**25.60**	8.30	14.40
bus:	15.90	23.50	**31.90**	**59.20**	16.70	24.50
train:	**6.80**	**16.70**	6.10	13.00	3.30	8.60
motor:	0.00	0.00	4.40	18.60	**28.00**	**60.00**
bicycle:	3.10	12.30	1.80	9.30	**3.80**	**15.10**
average:	7.03	15.30	10.94	25.73	**13.29**	**29.54**

(a) RGB & GT (b) No adaptation (c) Adapted Detections

Fig. 6: Samples from KITTI dataset. GT: Ground Truth, All instances are estimated after resize operation.

Fig. 7: Top-left: Normal condition, Top-right: Fog-02, Bottom-left: Fog-01 and Bottom-right: Fog-005.

TABLE II: Average Results for Three Fog Conditions

	Initial Model		Adapted Model	
(%)	**AP**	**AP 50%**	**AP**	**AP 50%**
Original	**10.9%**	**25.7%**	9.7%	24.6%
Fog-02	7.3%	16.2%	**7.4%**	**17.8%**
Fog-01	**9.1%**	**21.1%**	8.6%	21.0%
Fog-005	**9.6%**	22.5%	9.2%	**23.2%**

(a) RGB & GT (b) No adaptation (c) Adapted Detections

Fig. 8: Samples from CS Fog-02 dataset. GT: Ground Truth. Red circle shows important differences.

TABLE III: Detailed Results for Fog-02

Fog-02	**Initial Model**		**Adapted Model**	
(%)	**AP**	**AP 50%**	**AP**	**AP 50%**
person:	2.5%	11.3%	**3.0%**	**13.4%**
car:	12.8%	29.7%	**13.2%**	**31.3%**
truck:	8.9%	16.0%	**10.4%**	**18.2%**
bus:	**21.7%**	**34.3%**	19.6%	32.3%
train:	**1.3%**	**4.3%**	0.9%	**4.3%**
motorcycle:	2.3%	9.9%	**3.1%**	**14.7%**
bicycle:	1.4%	8.1%	**1.9%**	**10.2%**
average:	7.3%	16.2%	**7.4%**	**17.8%**

future work, we plan to train the backbone of our model on both datasets for new conditions to be able to compare its performance to supervised cases, propose an instance segmentation architecture for real-time inference and do domain adaptation for panoptic segmentation.

ACKNOWLEDGEMENT

Parts of the experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations.

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," *arXiv preprint arXiv:1708.02551*, 2017.

[4] C. Sakaridis, D. Dai, and L. V. Gool, "Semantic Foggy Scene Understanding with Synthetic Data," *IJCV*, pp. 1–20, 2018.

[5] 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.

[6] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning Transferable Features with Deep Adaptation Networks," in *ICML*, vol. 37, 2015.

[7] O. Erkent and C. Laugier, "Semantic segmentation with unsupervised domain adaptation under varying weather conditions for autonomous vehicles," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3580–3587, 2020.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," pp. 2961–2969, 2017.

[9] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, *et al.*, "Hybrid task cascade for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4974–4983.

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

[11] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2019, pp. 9157–9166.

[12] K. Sofiiuk, O. Barinova, and A. Konushin, "Adaptis: Adaptive instance selection network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7355–7363.

[13] T.-J. Yang, M. D. Collins, Y. Zhu, J.-J. Hwang, T. Liu, X. Zhang, V. Sze, G. Papandreou, and L.-C. Chen, "Deeperlab: Single-shot image parser," *arXiv preprint arXiv:1902.05093*, 2019.

[14] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[15] B. Romera-Paredes and P. H. S. Torr, "Recurrent instance segmentation," in *European conference on computer vision*. Springer, 2016, pp. 312–329.

[16] W. Xu, H. Wang, F. Qi, and C. Lu, "Explicit shape encoding for real-time instance segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5168–5177.

[17] S. Peng, W. Jiang, H. Pi, X. Li, H. Bao, and X. Zhou, "Deep snake for real-time instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8533–8542.

[18] D. Dai and L. Van Gool, "Dark Model Adaptation: Semantic Image Segmentation from Daytime to Nighttime," in *ITSC*, 2018.

[19] M. Wulfmeier, A. Bewley, and I. Posner, "Incremental Adversarial Domain Adaptation for Continually Changing Environments," in *ICRA*, 2018.

[20] D.-k. Kim, D. Maturana, M. Uenoyama, and S. Scherer, "Season-Invariant Semantic Segmentation with A Deep Multimodal Network," in *Field and Service Robotics*, 2018, pp. 255–270.

[21] Y. Chen, W. Li, X. Chen, and L. Van Gool, "Learning Semantic Segmentation from Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach," in *CVPR*, 2019.

[22] O. Erkent, C. Wolf, C. Laugier, D. Gonzalez, and V. Cano, "Semantic grid estimation with a hybrid bayesian and deep neural network approach," in *IEEE IROS*, 2018, pp. 888–895.

[23] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez, "ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation," in *CVPR*, 2019.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[25] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, vol. 17, no. 1, pp. 2096–2030, 2016.

[26] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker, "Learning to Adapt Structured Output Space for Semantic Segmentation," in *CVPR*, 2018, pp. 7472–7481.

[27] Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei, "Fully Convolutional Adaptation Networks for Semantic Segmentation," in *CVPR*, 2018, pp. 6810–6818.

[28] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, "Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation," in *CVPR*, 2018, pp. 3752–3761.

[29] H. Zhang, Y. Tian, K. Wang, H. He, and F.-Y. Wang, "Synthetic-to-real domain adaptation for object instance segmentation," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–7.

[30] X. Peng, Y. Li, Y. L. Murphey, X. Wei, and J. Luo, "Domain Adaptation with One-step Transformation," in *IEEE, SSCI*. IEEE, 2018, pp. 539–546.

[31] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[32] J. Hoffman, E. Tzeng, T. Park, J.-y. Zhu, U. C. Berkeley, P. Isola, K. Saenko, A. A. Efros, T. Darrell, and U. C. Berkeley, "CYCADA: Cycle-Consistent Adversarial Domain Adaptation," in *ICML*, 2018, pp. 1–15.

[33] Y. Zhang, P. David, and B. Gong, "Curriculum Domain Adaptation for Semantic Segmentation of Urban Scenes," in *ICCV*, 2018.

[34] Y. Li, L. Yuan, and N. Vasconcelos, "Bidirectional Learning for Domain Adaptation of Semantic Segmentation," in *CVPR*, 2019, pp. 6936–6945.

[35] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *CVPR*, 2017.

[36] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

[37] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NIPS*, 2017, pp. 5767–5777.

[38] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *IJCV*, 2018.

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[40] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016.