PolyTransform: Deep Polygon Transformer for Instance Segmentation

Justin Liang[1] Namdar Homayounfar[1,2]
Wei-Chiu Ma[1,3] Yuwen Xiong[1,2] Rui Hu[1] Raquel Urtasun[1,2]
[1]Uber Advanced Technologies Group [2]University of Toronto [3] MIT
{justin.liang,namdar,weichiu,yuwen,rui.hu,urtasun}@uber.com

Abstract

In this paper, we propose PolyTransform, a novel instance segmentation algorithm that produces precise, geometry-preserving masks by combining the strengths of prevailing segmentation approaches and modern polygon-based methods. In particular, we first exploit a segmentation network to generate instance masks. We then convert the masks into a set of polygons that are then fed to a deforming network that transforms the polygons such that they better fit the object boundaries. Our experiments on the challenging Cityscapes dataset show that our PolyTransform significantly improves the performance of the backbone instance segmentation network and ranks 1st on the Cityscapes test-set leaderboard. We also show impressive gains in the interactive annotation setting. [1]

1. Introduction

The goal of *instance segmentation* methods is to identify all countable objects in the scene, and produce a mask for each of them. With the help of instance segmentation, we can have a better understanding of the scene [68], design robotics systems that are capable of complex manipulation tasks [17], and improve perception systems of self-driving cars [44]. The task is, however, extremely challenging. In comparison to the traditional semantic segmentation task that infers the category of each pixel in the image, instance segmentation also requires the system to have the extra notion of individual objects in order to associate each pixel with one of them. Dealing with the wide variability in the scale and appearance of objects as well as occlusions and motion blur make this problem extremely difficult.

To address these issues, most modern instance segmentation methods employ a two-stage process [21, 63, 42], where object proposals are first created and then foreground background segmentation within each bounding box is performed. With the help of the box, they can better handle situations (*e.g.*, occlusions) where other methods often fail [4]. While these approaches have achieved state-of-the-art performance on multiple benchmarks (e.g., COCO [38], Cityscapes [11]) their output is often over-smoothed failing to capture fine-grained details.

An alternative line of work tackles the problem of *interactive annotation* [5, 2, 62, 39]. These techniques have been developed in the context of having an annotator in the loop, where a ground truth bounding box is provided. The goal of these works is to speed up annotation work by providing an initial polygon for annotators to correct as annotating from scratch is a very expensive process. In this line of work, methods exploit polygons to better capture the geometry of the object [5, 2, 39], instead of treating the problem as a pixel-wise labeling task. This results in more precise masks and potentially faster annotation speed as annotators are able to simply correct the polygons by moving the vertices. However, these approaches suffer in the presence of large occlusions or when the object is split into multiple disconnected components.

With these problems in mind, in this paper we develop a novel model, which we call *PolyTransform*, and tackle both the instance segmentation and interactive annotation problems. The idea behind our approach is that the segmentation masks generated by common segmentation approaches can be viewed as a starting point to compute a set of polygons, which can then be refined. We performed this refinement via a deforming network that predicts for each polygon the displacement of each vertex, taking into account the location of all vertices. By deforming each polygon, our model is able to better capture the local geometry of the object. Unlike [5, 2, 39], our model has no restriction on the number of polygons utilized to describe each object. This allows us to naturally handle cases where the objects are split into parts due to occlusion.

We first demonstrate the effectiveness of our approach on the Cityscapes dataset [11]. On the task of instance segmentation, our model improves the initialization by **3.0** AP and **10.3** in the boundary metric on the validation set. Importantly, we achieve **1st** place on the test set leaderboard, beating the current state of the art by **3.7** AP. We further eval-

[1]The supplementary of this paper can be found here.

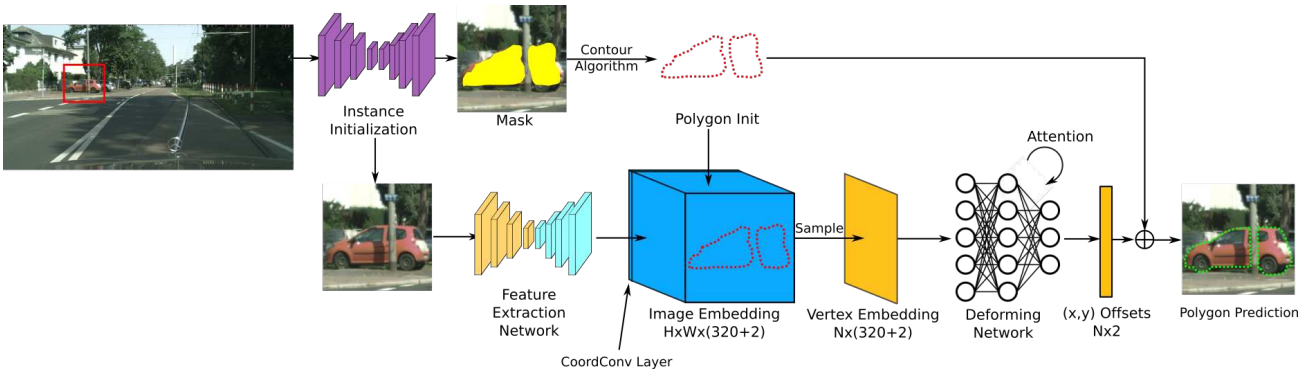

Figure 1. **Overview of our PolyTransform model.**

uate our model on a new self-driving dataset. Our model improves the initialization by **2.1** AP and **5.6** in the boundary metric. In the context of interactive annotation, we outperform the previous state of the art [62] by **2.0**% in the boundary metric. Finally, we conduct an experiment where the crowd-sourced labelers annotate the object instances using the polygon output from our model. We show that this can speed up the annotation time by **35**%!

2. Related Work

In this section, we briefly review the relevant literature on instance segmentation and annotation in the loop.

Proposal-based Instance Segmentation: Most modern instance segmentation models adopt a two-stage pipeline . First, an over-complete set of segment proposals is identified, and then a voting process is exploited to determine which one to keep [8, 14] As the explicit feature extraction process [53] is time-consuming [19, 20], Dai *et al*. [13, 12] integrated feature pooling into the neural network to improve efficiency. While the speed is drastically boosted comparing to previous approaches, it is still relatively slow as these approach is limited by the traditional detection based pipeline. With this problem in mind, researchers have looked into directly generating instance masks in the network and treat them as proposals [51, 52]. Based on this idea, Mask R-CNN [21] introduced a joint approach to do both mask prediction and recognition. It builds upon Faster R-CNN [54] by adding an extra parallel header to predict the object's mask, in addition to the existing branch for bounding box recognition. Liu *et al*. [42] propose a path aggregation network to improve the information flow in Mask R-CNN and further improve performance. More recently, Chen *et al*. [6] interleaves bounding box regression, mask regression and semantic segmentation together to boost instance segmentation performance. Xu *et al*. [64] fit Chebyshev polynomials to instances by having a network learn the coefficients, this allows for real time instance segmentation. Huang *et al*. [25] optimize the scoring of the bounding boxes by predicting IoU for each mask rather than only a classification score. Kuo *et al*. [34] start with bounding boxes and refine them using shape priors. Xiong *et al*. [63] and Kirillov *et al*. [31] extended Mask R-CNN to the task of panoptic segmentation. Yang *et al*. [65] extended Mask R-CNN to the task of video instance segmentation.

Proposal-free Instance Segmentation: This line of research aims at segmenting the instances in the scene without an explicit object proposal. Zhang *et al*. [67, 66] first predicts instance labels within the extracted multi-scale patches and then exploits dense Conditional Random Field [33] to obtain a consistent labeling of the full image. While achieving impressive results, their approach is computationally intensive. Bai and Urtasun [4] exploited a deep network to predict the energy of the watershed transform such that each basin corresponds to an object instance. With one simple cut, they can obtain the instance masks of the whole image without any post-processing. Similarly, [32] exploits boundary prediction to separate the instances within the same semantic category. Despite being much faster, they suffer when dealing with far or small objects whose boundaries are ambiguous. To address this issue, Liu *et al*. [41] present a sequential grouping approach that employs neural networks to gradually compose objects from simpler elements. It can robustly handle situations where a single instance is split into multiple parts. Newell and Deng [49] implicitly encode the grouping concept into neural networks by having the model to predict both semantic class and a tag for each pixel. The tags are one dimensional embeddings which associate each pixel with one another. Kendall *et al*. [28] propose a method to assign pixels to objects having each pixel point to its object's center so that it can be grouped. Sofiiuk *et al*. [58] use a point proposal network to generate points where the instances can be, this is then processed by a CNN to outputs instance masks for each location. Neven *et al*. [48] propose a new clustering loss that pulls the spatial embedding of pixels belonging to the same instance together to achieve real time instance segmentation while having high accuracy. Gao *et al*. [18] propose a single shot instance segmentation network that outputs a pixel pair affinity pyramid to compute whether two pixels belong to the same instance, they then combine this with a predicted semantic segmentation to output a single instance

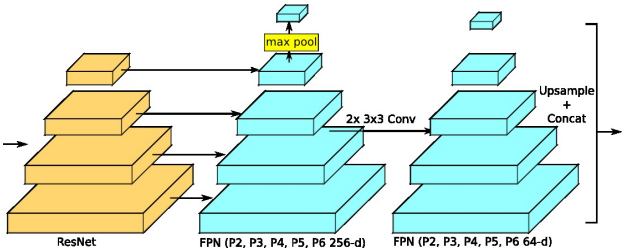

Figure 2. **Our feature extraction network.**

segmentation map.

Interactive Annotation: The task of interactive annotation can also be posed as finding the polygons or curves that best fit the object boundaries. In fact, the concept of deforming a curve to fit the object contour can be dated back to the 80s where the active contour model was first introduced [27]. Since then, variants of ACM [10, 47, 9] have been proposed to better capture the shape. Recently, the idea of exploiting polygons to represent an instance is explored in the context of human in the loop segmentation [5, 2]. Castrejón *et al*. [5] adopted an RNN to sequentially predict the vertices of the polygon. Acuna *et al*. [2] extended [5] by incorporating graph neural networks and increasing image resolution. While these methods demonstrated promising results on public benchmarks [11], they require ground truth bounding box as input. Ling *et al*. [39] and Dong *et al*. [16] exploited splines as an alternative parameterization. Instead of drawing the whole polygon/curves from scratch, they start with a circle and deform it. Wang *et al*. tackled this problem with implicit curves using level sets [62], however, because the outputs are not polygons, an annotator cannot easily corrected them. In [46], Maninis *et al*. use extreme boundary as inputs rather than bounding boxes and Majumder *et al*. [45] uses user clicks to generate content aware guidance maps; all of these help the networks learn stronger cues to generate more accurate segmentations. However, because they are pixel-wise masks, they are not easily amenable by an annotator. Acuna *et al*. [1] develop an approach that can be used to refine noisy annotations by jointly reasoning about the object boundaries with a CNN and a level set formulation. In the domain of offline mapping, several papers from Homayounfar *et al*. and Liang *et al*. [23, 35, 24, 36] have tackled the problem of automatically annotating crosswalks, road boundaries and lanes by predicting structured outputs such as a polyline.

3. PolyTransform

Our aim is to design a robust segmentation model that is capable of producing precise, geometry-preserving masks for each individual object. Towards this goal, we develop PolyTransform, a novel deep architecture that combines prevailing segmentation approaches [21, 63] with modern polygon-based methods [5, 2]. By exploiting the best of both worlds, we are able to generate high quality segmentation masks under various challenging scenarios.

In this section, we start by describing the backbone architecture for feature extraction and polygon initialization. Next, we present a novel deforming network that warps the initial polygon to better capture the local geometry of the object. An overview of our approach is shown in Figure 1.

3.1. Instance Initialization

The goal of our instance initialization module is to provide a good polygon initialization for each individual object. To this end, we first exploit a model to generate a mask for each instance in the scene. Our experiments show that our approach can significantly improve performance for a wide variety of segmentation models. If the segmentation model outputs proposal boxes, we use them to crop the image, otherwise, we fit a bounding box to the mask. The cropped image is later resized to a square and fed into a feature network (described in Sec. 3.2) to obtain a set of reliable deep features. In practice, we resize the cropped image to $(H_c, W_c) = (512, 512)$. To initialize the polygon, we use the border following algorithm of [60] to extract the contours from the predicted mask. We get the initial set of vertices by placing a vertex at every 10 px distance in the contour. Empirically, we find such dense vertex interpolation provides a good balance between performance and memory consumption.

3.2. Feature Extraction Network

The goal of our feature extraction network is to learn strong object boundary features. This is essential as we want our polygons to capture high curvature and complex shapes. As such, we employ a feature pyramid network (FPN) [37] to learn and make use of multi-scale features. This network takes as input the (H_c, W_c) crop obtained from the instance initialization stage and outputs a set of features at different pyramid levels. Our backbone can be seen in Figure 2.

3.3. Deforming Network

We have computed a polygon initialization and deep features of the FPN from the image crop. Next we build a feature embedding for all N vertices and learn a deforming model that can effectively predict the offset for each vertex so that the polygon snaps better to the object boundaries.

Vertex embedding: We build our vertex representation upon the multi-scale feature extracted from the backbone FPN network of the previous section. In particular, we take the P_2, P_3, P_4, P_5 and P_6 feature maps and apply two lateral convolutional layers to each of them in order to reduce the number of feature channels from 256 to 64 each. Since the feature maps are $1/4$, $1/8$, $1/16$, $1/32$ and $1/64$ of

	training data	AP_{val}	AP	AP_{50}	person	rider	car	truck	bus	train	mcycle	bcycle
DWT [4]	fine	21.2	19.4	35.3	15.5	14.1	31.5	22.5	27.0	22.9	13.9	8.0
Kendall et al. [28]	fine	–	21.6	39.0	19.2	21.4	36.6	18.8	26.8	15.9	19.4	14.5
Arnab et al. [3]	fine	–	23.4	45.2	21.0	18.4	31.7	22.8	31.1	31.0	19.6	11.7
SGN [41]	fine+coarse	29.2	25.0	44.9	21.8	20.1	39.4	24.8	33.2	30.8	17.7	12.4
PolygonRNN++ [2]	fine	–	25.5	45.5	29.4	21.8	48.3	21.2	32.3	23.7	13.6	13.6
Mask R-CNN [21]	fine	31.5	26.2	49.9	30.5	23.7	46.9	22.8	32.2	18.6	19.1	16.0
BShapeNet+ [29]	fine	–	27.3	50.4	29.7	23.4	46.7	26.1	33.3	24.8	20.3	14.1
GMIS [43]	fine+coarse	–	27.3	45.6	31.5	25.2	42.3	21.8	37.2	28.9	18.8	12.8
Neven et al. [48]	fine	–	27.6	50.9	34.5	26.1	52.4	21.7	31.2	16.4	20.1	18.9
PANet [42]	fine	36.5	31.8	57.1	36.8	30.4	54.8	27.0	36.3	25.5	22.6	20.8
Mask R-CNN [21]	fine+COCO	36.4	32.0	58.1	34.8	27.0	49.1	30.1	40.9	30.9	24.1	18.7
AdaptIS [58]	fine	36.3	32.5	52.5	31.4	29.1	50.0	31.6	41.7	39.4	24.7	12.1
SSAP [18]	fine	37.3	32.7	51.8	35.4	25.5	55.9	33.2	43.9	31.9	19.5	16.2
BShapeNet+ [29]	fine+COCO	–	32.9	58.8	36.6	24.8	50.4	33.7	41.0	33.7	25.4	17.8
UPSNet [63]	fine+COCO	37.8	33.0	59.7	35.9	27.4	51.9	31.8	43.1	31.4	23.8	19.1
PANet [42]	fine+COCO	41.4	36.4	63.1	41.5	33.6	58.2	31.8	45.3	28.7	28.2	**24.1**
Ours	fine+COCO	**44.6**	**40.1**	**65.9**	**42.4**	**34.8**	**58.5**	**39.8**	**50.0**	**41.3**	**30.9**	23.4

Table 1. **Instance segmentation on Cityscapes val and test set:** This table shows our instance segmentation results on Cityscape test. We report models trained on fine and fine+COCO. We report AP and AP_{50}.

	fine	COCO	AP	AP_{50}	car	truck	bus	train	person	rider	bcycle+r	bcycle	mcycle+r	mcycle
Mask RCNN [21]	✓	-	26.6	53.5	47.0	41.1	42.8	10.7	32.8	27.5	18.6	10.2	14.8	20.2
PANet [42]	✓	-	26.6	53.5	46.6	41.8	44.2	2.7	32.8	27.4	18.7	11.3	15.1	25.8
UPSNet [63]	✓	-	29.0	56.0	47.1	41.8	47.8	12.7	33.5	27.3	18.6	10.4	20.4	30.2
PANet [42]	✓	✓	29.1	55.2	47.4	43.7	47.6	10.7	34.4	30.1	20.5	11.8	17.3	27.4
UPSNet [63]	✓	✓	31.5	58.4	46.9	44.0	49.8	21.6	34.1	30.3	21.7	12.8	19.3	**34.5**
Ours	✓	✓	**35.3**	**60.8**	**50.5**	**47.3**	**52.5**	**23.4**	**40.4**	**37.0**	**25.1**	**16.0**	**28.7**	32.6

Table 2. **Instance segmentation on test set of our new self-driving dataset:** This table shows our instance segmentation results our new dataset's test set. We report models trained on fine and fine+COCO. We report AP and AP_{50}. +r is short for with rider.

the original scale, we bilinearly upsample them back to the original size and concatenate them to form a $H_c \times W_c \times 320$ feature tensor. To provide the network a notion of where each vertex is, we further append a 2 channel CoordConv layer [40]. The channels represent x and y coordinates with respect to the frame of the crop. Finally, we exploit the bilinear interpolation operation of the spatial transformer network [26] to sample features at the vertex coordinates of the initial polygon from the feature tensor. We denote such $N \times (320 + 2)$ embedding as $\boldsymbol{z}$.

Deforming network: When moving a vertex in a polygon, the two attached edges are subsequently moved as well. The movement of these edges depends on the position of the neighboring vertices. Each vertex thus must be aware of its neighbors and needs a way to communicate with one another in order to reduce unstable and overlapping behavior. In this work, we exploit the self-attending Transformer network [61] to model such intricate dependencies. We leverage the attention mechanism to propagate the information across vertices and improve the predicted offsets. More formally, given the vertex embeddings $\boldsymbol{z}$, we first employ three feed-forward neural networks to transform it into $Q(\boldsymbol{z})$, $K(\boldsymbol{z})$, $V(\boldsymbol{z})$, where Q, K, V stands for Query, Key and Value. We then compute the weightings between vertices by taking a softmax over the dot product $Q(\boldsymbol{z})K(\boldsymbol{z})^T$. Finally, the weightings are multiplied with the keys $V(\boldsymbol{z})$ to propagate these dependencies across all vertices. Such attention mechanism can be written as:

$$Atten(Q(\boldsymbol{z}), K(\boldsymbol{z}), V(\boldsymbol{z})) = softmax(\frac{Q(\boldsymbol{z})K(\boldsymbol{z})^T}{\sqrt{d_k}})V(\boldsymbol{z}),$$

where d_k is the dimension of the queries and keys, serving as a scaling factor to prevent extremely small gradients. We repeat the same operation a fixed number of times, 6 in our experiments. After the last Transformer layer, we feed the output to another feed-forward network which predicts $N \times 2$ offsets for the vertices. We add the offsets to the polygon initialization to transform the shape of the polygon.

3.4. Learning

We train the deforming network and the feature extraction network in an end-to-end manner. Specifically, we minimize the weighted sum of two losses. The first penalizes the model for when the vertices deviate from the ground truth. The second regularizes the edges of the polygon to prevent overlap and unstable movement of the vertices.

Init	Backbone	COCO	AP	AP_{gain}	AF	AF_{gain}
DWT	Res101	-	18.7	+2.2	44.2	+5.8
UPSNet	Res50	-	33.3	+3.0	41.4	+10.3
UPSNet	Res50	✓	37.8	+2.4	45.7	+7.8
UPSNet	WRes38+PANet	✓	41.4	+1.6	51.1	+4.9
UPSNet	WRes38+PANet+DCN	✓	43.0	+1.6	51.5	+4.2

Table 3. **Improvement on Cityscapes val instance segmentation initializations:** We report the AP, AF of the initialization and gain in AP, AF from the initialization instances when running our PolyTransform model for Cityscapes val.

Init	Backbone	COCO	AP	AP_{gain}	AF	AF_{gain}
M-RCNN	Res50	-	28.8	+2.2	44.2	+5.6
UPSNet	Res101	-	31.7	+1.6	45.7	+3.2
UPSNet	Res101	✓	34.2	+1.9	45.8	+3.4
UPSNet	WRes38+PANet+DCN	✓	36.1	+1.4	50.1	+3.4

Table 4. **Improvement over instance segmentation initializations on the validation of our new self-driving dataset:** We report the AP, AF of the initialization and gain in AP, AF from the initialization instances when running our PolyTransform model for the validation of our new self-driving dataset.

Polygon Transforming Loss: We make use of the Chamfer Distance loss similar to [23] to move the vertices of our predicted polygon P closer to the ground truth polygon Q. The Chamfer Distance loss is defined as:

$$L_c(P,Q) = \frac{1}{|P|}\sum_i \min_{q\in Q} \|p_i - q\|_2 + \frac{1}{|Q|}\sum_j \min_{p\in P} \|p - q_j\|_2$$

where p and q are the rasterized edge pixels of the polygons P and Q. To prevent unstable movement of the vertices, we add a deviation loss on the lengths of the edges $\boldsymbol{e}$ between vertices. Empirically, we found that without this term the vertices can suddenly shift a large distance, incurring a large loss and causing the gradients to blow up. We define the standard deviation loss as: $L_s(P) = \sqrt{\frac{\sum\|\boldsymbol{e}-\bar{\boldsymbol{e}}\|_2}{n}}$, where $\bar{\boldsymbol{e}}$ denotes the mean length of the edges.

4. Experiments

We evaluate our model in the context of both instance segmentation and interactive annotation settings.

Experimental Setup: We train our model on 8 Titan 1080 Ti GPUs using the distributed training framework Horovod [56] for 1 day. We use a batch size of 1, ADAM [30], 1e-4 learning rate and a 1e-4 weight decay. We augment our data by randomly flipping the images horizontally. During training, we only train with instances whose proposed box has an Intersection over Union (IoU) overlap of over 0.5 with the ground truth (GT) boxes. We train with both instances produced using proposed boxes and GT boxes to further augment the data. For our instance segmentation experiments, we augment the box sizes by -3% to $+3\%$ during training and test with a 2% box expansion. For our interactive annotation experiments, we train and test on boxes with an expansion of 5px on each side; we only compute a chamfer loss if the predicted vertex is at least 2px from the ground truth polygon. When placing weights on the losses, we found ensuring the loss values were approximately balanced produced the best result. For our PolyTransform FPN, we use ResNet50 [22] as the backbone and use the same pretrained weights from UPSNet [63] on Cityscapes. For our deforming network we do not use pretrained weights.

Cityscapes (fine+COCO)	AP	AF
UPSNet	43.0	51.5
Baseline 1	43.8	52.6
Baseline 2	43.5	52.4
Ours	**44.6**	**55.7**

Table 5. **Comparison with naive refiners on Cityscapes val set.**

4.1. Instance Segmentation

Datasets: We use Cityscapes [11] which has high quality pixel-level instance segmentation annotations. The 1024×2048 images were collected in 27 cities, and they are split into 2975, 500 and 1525 images for train/val/test. There are 8 instance classes: bicycle, bus, person, train, truck, motorcycle, car and rider. We also report results on a new dataset we collected. It consists of 10235/1139/1186 images for train/val/test split annotated with 10 classes: car, truck, bus, train , person, rider, bicycle with rider, bicycle, motorcycle with rider and motorcycle. Each image is of size 1200×1920.

Metrics: For our instance segmentation results, we report the average precision (AP and AP_{50}) for the predicted mask. Here, the AP is computed at 10 IoU overlap thresholds ranging from 0.5 to 0.95 in steps of 0.05 following [11]. AP_{50} is the AP at an overlap of 50%. We also introduce a new metric that focusses on boundaries. In particular, we use a metric similar to [62, 50] where a precision, recall and F1 score is computed for each mask, where the prediction is correct if it is within a certain distance threshold from the ground truth. We use a threshold of 1px, and only compute the metric for true positives. We use the same 10 IoU overlap thresholds ranging from 0.5 to 0.95 in steps of 0.05 to determine the true positives. Once we compute the F1 score for all classes and thresholds, we take the average over all examples to get AF.

Instance Initialization: We want to use a strong instance initialization to show that we can still improve the results. We take the publicly available UPSNet [63], and replace its backbone with WideResNet38 [55] and add all the elements of PANet [42] except for the synchronized batch normalization (we use group normalization instead). We then pretrain on COCO and use deformable convolution (DCN) [15] in the backbone.

Comparison to SOTA: As shown in 1 we outperform all baselines in every metric on the val and test sets of Cityscapes. We achieve a new state-of-the-art test result

Input Image | Our Instance Segmentation | GT Instance Segmentation

Figure 3. We showcase qualitative instance segmentation results of our model on the Cityscapes validation set.

of 40.1AP. This outperforms PANet by 3.7 and 2.8 points in AP and AP_{50m} respectively. It also ranks number 1 on the official Cityscapes leaderboard. We report the results on our new dataset in Table 2. We achieve the strongest test AP result in this leaderboard. We see that we improve over PANet by 6.2 points in AP and UPSNet by 3.8 points in AP.

Robustness to Initialization: We report the improvement over different instance segmentation networks used as initialization in Table 3 on Cityscapes, showing significant and consistent improvements in val AP across all models. When we train our model on top of the DWT [4] instances we see an improvement of $+2.2$, $+5.8$ points in AP and AF. We also train on top of the UPSNet results from the original paper along with UPSNet with `WRes38+PANet` as a way to reproduce the current SOTA val AP of PANet. We show an improvement of $+1.6$, $+4.9$ points in AP and AF. Finally we improve on our best initialization by $+1.6, +4.2$ AP points in AP and AF. As we can see, our boundary metric sees a very consistent $4\% - 10\%$ gain in AF across all models. This suggests that our approach significantly improvs the instances at the boundary. We notice that a large gain in AP (`WRes38+PANet` to `WRes38+PANet+DCN`) does not necessarily translate to a large gain in AF, however, our model will always provide a significant increase in this metric. We also report the validation AP improvement over different instance segmentation networks in Table 4 for our new dataset. We see that we can improve on Mask R-CNN [21] by $+2.2$, $+5.6$ points in AP, AF. For the different UPSNet models, we improve upon it between 1.4-2.2 AP points. Once again, our model shows a consistent and strong improvement over all initializations. We also see a very consistent $3\% - 6\%$ gain in AF across all the models.

Annotation Efficiency: We conduct an experiment where we ask crowd-sourced labelers to annotate 150 images from our new dataset with instances larger than 24x24px for vehicles and 12x14px for pedestrians/riders. We performed a control experiment where the instances are annotated completely from scratch (without our method) and a parallel experiment where we use our model to output the instances for them to fix to produce the final annotations. In the fully manual experiment, it took on average 60.3 minutes to annotate each image. When the annotators were given the PolyTransform output to annotate on top of, it took on average 39.4 minutes to annotate each image. Thus reducing 35% of the time required to annotate the images. This resulted in significant cost savings.

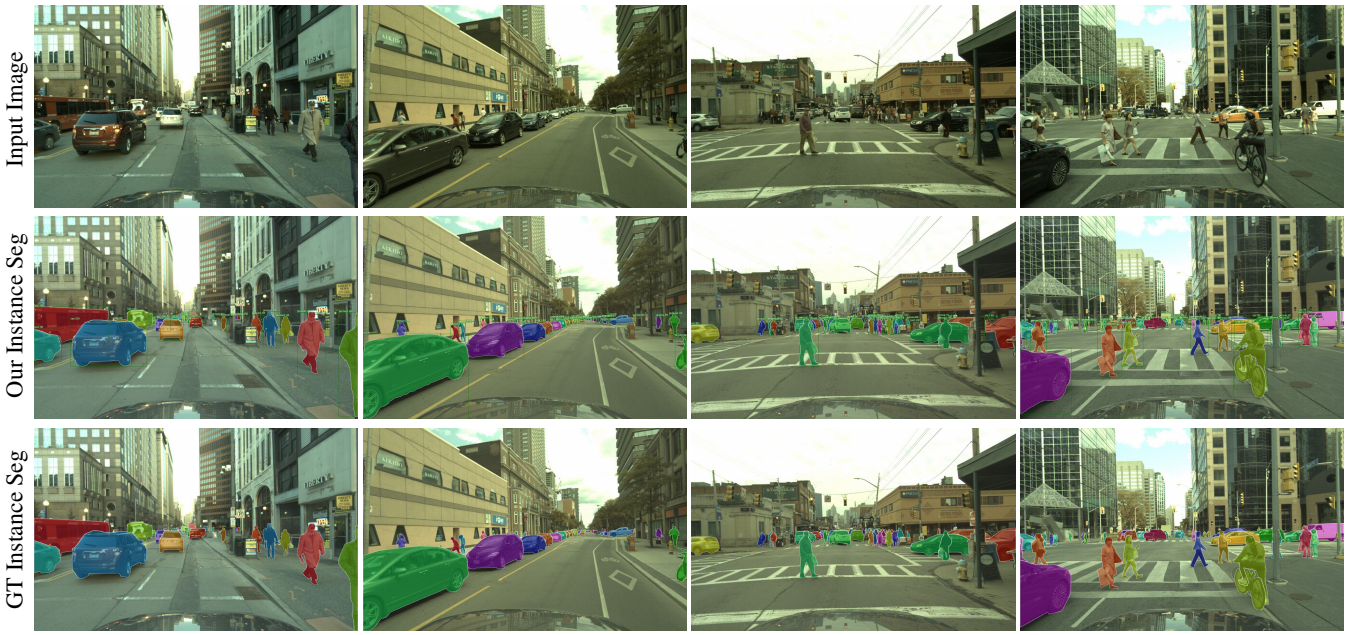

Figure 4. We showcase the qualitative instance segmentation results of our model on the validation set of our new self-driving dataset

	Mean	bicycle	bus	person	train	truck	mcycle	car	rider	F_{1px}	F_{2px}
DEXTR* [46]	79.11	71.92	87.42	78.36	78.11	84.88	72.41	84.62	75.18	54.00	68.60
Deep Level Sets [62]	80.86	**74.32**	**88.85**	80.14	**80.35**	86.05	74.10	86.35	76.74	60.29	74.40
Ours	**80.90**	74.22	88.78	**80.73**	77.91	**86.45**	**74.42**	**86.82**	**77.85**	**62.33**	**76.55**

Table 6. **Interactive Annotation (Cityscapes Stretch):** This table shows our IoU % performance in the setting of annotation where we are given the ground truth boxes. DEXTR* represents DEXTR without extreme points.

	Mean	bicycle	bus	person	train	truck	mcycle	car	rider	F_{1px}	F_{2px}
Polygon-RNN [5]	61.40	52.13	69.53	63.94	53.74	68.03	52.07	71.17	60.58	–	–
Polygon-RNN++ [2]	71.38	63.06	81.38	72.41	64.28	78.90	62.01	79.08	69.95	46.57	62.26
Curve GCN [39]	73.70	67.36	85.43	73.72	64.40	80.22	64.86	81.88	71.73	47.72	63.64
Deep Level Sets [62]	73.84	67.15	83.38	73.07	69.10	80.74	65.29	81.08	70.86	48.59	64.45
Ours	**78.76**	**72.97**	**87.53**	**78.58**	**72.25**	**85.08**	**72.50**	**85.36**	**75.83**	**56.89**	**71.60**

Table 7. **Interactive Annotation (Cityscapes Hard):** This table shows our IoU % performance in the setting of annotation where we are given the ground truth boxes.

Naive refiner: We implemented two baselines that apply a semantic segmentation network on top of the initial mask. 1) We replace PolyTransform with a refinement network inspired by DeepLabV3 [7] and PWC-Net [59] . It takes as input the same initialization mask, the cropped RGB image and the cropped feature, and exploits a series of convolutions to refine the binary mask. 2) We add an extra head to UPSNet, with the initialization mask and the cropped feature as input to refine the binary mask. The head's architecture is similar to that of the semantic head (i.e., uses the features from UPSNet's FPN). For fairness, the number of parameters of both baselines are similar to PolyTransform. As shown in Tab. 5, our approach performs the best.

Timing: Our model takes 575 ms to process each image on Cityscapes. This can easily be improved with more GPU memory, as this will allow to batch all the instances. Furthermore, the hidden dimension of the FPN can be tuned to speed up the model.

Qualitative Results: We show qualitative results of our model on the validation set in Figure 3. In our instance segmentation outputs we see that in many cases our model is able to handle occlusion. For example, in row 3, we see that our model is able to capture the feet of the purple and blue pedestrians despite their feet being occluded from the body. We also show qualitative results on our new dataset in Figure 4. We see that our model is able to capture precise boundaries, allowing it to capture difficult shapes such as car mirrors and pedestrians.

Failure Modes: Our model can fail when the initialization is poor (left image in Figure 5). Despite being able to handle occlusion, our model can still fail when the occlusion is complex or ambiguous as seen in the right of Figure 5. Here there is a semi-transparent fence blocking the car.

	BBone	COCO	mIOU	mIOU_{gain}	F_1	$F_{1,gain}$	F_2	$F_{2,gain}$
FCN	R50	-	79.93	+0.15	59.43	+1.53	73.64	+1.30
FCN	R101	-	80.94	+0.11	60.64	+1.14	74.78	+1.06
FCN	R101	✓	80.65	+0.08	59.21	+1.39	73.47	+1.10
DeepLabV3	R50	-	80.41	+0.17	59.70	+1.51	73.81	+1.48
DeepLabV3	R101	-	80.93	+0.09	60.50	+1.18	74.44	+1.33
DeepLabV3+	R101	✓	80.90	+0.08	61.10	+1.23	75.25	+1.30

Table 8. **Improvement on Cityscapes Stretch segmentation initializations:** We report the metric improvements when running our PolyTransform model on different models. We report our model results trained on FCN [57] and DeepLabV3 [7]. DeepLabV3+ uses the class balancing loss from [46]. We report on models with various backbones (Res50 vs Res101) and also with and without pretraining on COCO [38].

4.2. Interactive Annotation

The goal is to annotate an object with a polygon given its ground truth bounding box. The idea is that the annotator provides a ground truth box and our model works on top of it to output a polygon representation of the object instance.

Dataset: We follow [5] and split the Cityscapes dataset such that the original val set is the test set and two cities from the training (Weimar and Zurich) form the val set. [62] further splits this dataset into two settings: 1) Cityscapes Hard, where the ground truth bounding box is enlarged to form a square and then the image is cropped. 2) Cityscapes Stretch, where the ground truth bounding box along with the image is stretched to a square and then cropped.

Metric: To evaluate our model for this task, we report the intersection over union (IoU) on a per-instance basis and average for each class. Then, following [5] this is averaged across all classes. We also report the boundary metric reported in [62, 50], which computes the F measure along the contour for a given threshold. The thresholds used are 1 and 2 pixels as Cityscapes contains a lot of small instances.

Instance Initialization: For our best model we use a variation of DeepLabV3 [7], which we call DeepLabV3+ as the instance initialization network. The difference is that we train DeepLabV3 with the class balancing loss used in [46].

Comparison to SOTA: Tables 6 and 7 show results on the test set in both Cityscapes Stretch and Hard. For Cityscapes Stretch, we see that our model significantly outperforms the SOTA in the boundary metric, improving it by up to 2%. Unlike the Deep Level Sets [62] method which outputs a pixel wise mask, our method outputs a polygon which allows for it to be amenable to modification by an annotator by simply moving the vertices. For Cityscapes Hard, our model outperforms the SOTA by 4.9%, 8.3% and 7.2% in mean IOU, F at 1px and F at 2px respectively.

Figure 5. **Failure modes:** (Left) Our model fails because the initialization is poor. (Right) The model fails because of complex occlusion. (Yellow: Initialization; Cyan: Ours)

Robustness to Initalization: We also report improvements over different segmentation initializations in Table 8, the results are on the test set. Our models are trained on various backbone initialization models (FCN [57] and DeepLabV3 [7] with and without pretraining on COCO [38]). Our model is able to consistently and significantly improve the boundary metrics at 1 and 2 pixels by up to 1.5% and we improve the IOU between 0.1-0.2%. We also note that the difference in mean IOU between FCN and DeepLabV3 is very small (at most 0.5%) despite DeepLabV3 being a much stronger segmentation model. We argue that the margin for mean IOU improvement is very small for this dataset.

Timing: Our model runs on average 21 ms per object instance. This is 14x faster than Polygon-RNN++ [2] and 1.4x faster than Curve GCN [39] which are the state of the arts.

5. Conclusion

In this paper, we present PolyTransform, a novel deep architecture that combines the strengths of both prevailing segmentation approaches and modern polygon-based methods. We first exploit a segmentation network to generate a mask for each individual object. The instance mask is then converted into a set of polygons and serve as our initialization. Finally, a deforming network is applied to warp the polygons to better fit the object boundaries. We evaluate the effectiveness of our model on the Cityscapes dataset as well as a novel dataset that we collected. Experiments show that our approach is able to produce precise, geometry-preserving instance segmentation that significantly outperforms the backbone model. Comparing to the instance segmentation initialization, we increase the validation AP and boundary metric by up to 3.0 and 10.3 points, allowing us to achieve 1st place on the Cityscapes leaderboard. We also show that our model speeds up annotation by 35%. Comparing to previous work on annotation-in-the-loop [2], we outperform the boundary metric by 2.0%. Importantly, our PolyTransform generalizes across various instance segmentation network.

References

[1] David Acuna, Amlan Kar, and Sanja Fidler. Devil is in the edges: Learning semantic boundaries from noisy annotations. In *CVPR*, 2019.

[2] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. 2018.

[3] Anurag Arnab and Philip H. S. Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *CVPR*, 2017.

[4] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017.

[5] Lluıs Castrejón, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017.

[6] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *CVPR*, 2019.

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, 2017.

[8] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *CVPR*, 2018.

[9] Dominic Cheng, Renjie Liao, Sanja Fidler, and Raquel Urtasun. DARNet: Deep active ray network for building segmentation. In *CVPR*, 2019.

[10] Laurent D Cohen. On active contour models and balloons. *CVGIP*, 1991.

[11] Marius Cordts, Mohamed Omran Sebastian Ramos, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele, Daimler Ag R, Tu Darmstadt, Mpi Informatics, and Tu Dresden. The cityscapes dataset.

[12] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016.

[13] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015.

[14] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

[15] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *ICCV*, 2017.

[16] Zihao Dong, Ruixun Zhang, and Xiuli Shao. Automatic annotation and segmentation of object instances with deep active curve network. *IEEE Access*, 2019.

[17] Nima Fazeli, Miquel Oller, Jiajun Wu, Zheng Wu, Joshua B. Tenenbaum, and Alberto Rodriguez. See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion. *Science Robotics*, 2019.

[18] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *ICCV*, 2019.

[19] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[20] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, 2015.

[23] Namdar Homayounfar, Wei-Chiu Ma, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. Hierarchical recurrent attention networks for structured online maps. In *CVPR*, 2018.

[24] Namdar Homayounfar, Wei-Chiu Ma, Justin Liang, Xinyu Wu, Jack Fan, and Raquel Urtasun. Dagmapper: Learning to map by discovering lane topology. In *ICCV*, 2019.

[25] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask Scoring R-CNN. In *CVPR*, 2019.

[26] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015.

[27] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *IJCV*, 1988.

[28] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

[29] Ha Young Kim and Ba Rom Kang. Instance segmentation and object detection with bounding shape masks. *CoRR*, 2018.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2014.

[31] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic segmentation. In *CVPR*, 2019.

[32] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from edges to instances with multicut. In *CVPR*, 2017.

[33] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

[34] Weicheng Kuo, Anelia Angelova, Jitendra Malik, and Tsung-Yi Lin. Shapemask: Learning to segment novel objects by refining shape priors. In *ICCV*, 2019.

[35] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Shenlong Wang, and Raquel Urtasun. Convolutional recurrent network for road boundary extraction. In *CVPR*, 2019.

[36] Justin Liang and Raquel Urtasun. End-to-end deep structured models for drawing crosswalks. In *ECCV*, 2018.

[37] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, 2016.

[38] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[39] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, 2019.

[40] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *CoRR*, 2018.

[41] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *ICCV*, 2017.

[42] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.

[43] Yiding Liu, Siyu Yang, Bin Li, Wengang Zhou, Jizheng Xu, Houqiang Li, and Yan Lu. Affinity derivation and graph merge for instance segmentation. In *ECCV*, 2018.

[44] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *CVPR*, 2019.

[45] Soumajit Majumder and Angela Yao. Content-aware multi-level guidance for interactive instance segmentation. In *CVPR*, 2019.

[46] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. *CVPR*, 2017.

[47] Diego Marcos, Devis Tuia, Benjamin Kellenberger, Lisa Zhang, Min Bai, Renjie Liao, and Raquel Urtasun. Learning deep structured active contours end-to-end. In *CVPR*, 2018.

[48] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *CVPR*, 2019.

[49] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *NIPS*, 2017.

[50] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus H. Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. *CVPR*, 2016.

[51] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *NIPS*, 2015.

[52] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *ECCV*, 2016.

[53] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *PAMI*, 2017.

[54] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[55] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *CVPR*, 2018.

[56] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *CoRR*, 2018.

[57] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[58] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *ICCV*, 2019.

[59] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. 2018.

[60] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 1985.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, 2017.

[62] Zian Wang, Huan Ling, David Acuna, Amlan Kar, and Sanja Fidler. Object instance annotation with deep extreme level set evolution. In *CVPR*, 2019.

[63] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. *CoRR*, 2019.

[64] Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *ICCV*, 2019.

[65] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *ICCV*, 2019.

[66] Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *CVPR*, 2016.

[67] Ziyu Zhang, Alexander G Schwing, Sanja Fidler, and Raquel Urtasun. Monocular object instance segmentation and depth ordering with cnns. In *ICCV*, 2015.

[68] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.