Deep Snake for Real-Time Instance Segmentation

Sida Peng[1] Wen Jiang[1] Huaijin Pi[1] Xiuli Li[2] Hujun Bao[1] Xiaowei Zhou[1]*
[1]Zhejiang University [2]Deepwise AI Lab

Abstract

This paper introduces a novel contour-based approach named deep snake for real-time instance segmentation. Unlike some recent methods that directly regress the coordinates of the object boundary points from an image, deep snake uses a neural network to iteratively deform an initial contour to match the object boundary, which implements the classic idea of snake algorithms with a learning-based approach. For structured feature learning on the contour, we propose to use circular convolution in deep snake, which better exploits the cycle-graph structure of a contour compared against generic graph convolution. Based on deep snake, we develop a two-stage pipeline for instance segmentation: initial contour proposal and contour deformation, which can handle errors in object localization. Experiments show that the proposed approach achieves competitive performances on the Cityscapes, KINS, SBD and COCO datasets while being efficient for real-time applications with a speed of 32.3 fps for 512×512 images on a 1080Ti GPU. The code is available at https://github.com/zju3dv/snake/.

1. Introduction

Instance segmentation is the cornerstone of many computer vision tasks, such as video analysis, autonomous driving, and robotic grasping, which require both accuracy and efficiency. Most of the state-of-the-art instance segmentation methods [18, 27, 5, 19] perform pixel-wise segmentation within a bounding box given by an object detector [36], which may be sensitive to the inaccurate bounding box. Moreover, representing an object shape as dense binary pixels generally results in costly post-processing.

An alternative shape representation is the object contour, which is a set of vertices along the object silhouette. In contrast to pixel-based representation, a contour is not limited within a bounding box and has fewer parameters. Such a contour-based representation has long been used in image segmentation since the seminal work by Kass et al. [21], which is well known as snakes or active contours. Given an initial contour, the snake algorithm iteratively deforms it to match the object boundary by optimizing an energy functional defined with low-level features, such as image intensity or gradient. While many variants [6, 7, 15] have been developed in literature, these methods are prone to local optima as the objective functions are handcrafted and typically nonconvex.

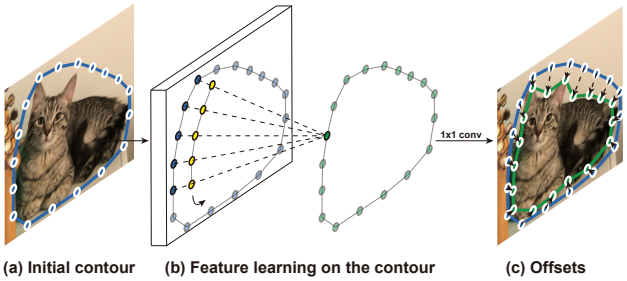

Figure 1. **The basic idea of deep snake.** Given an initial contour, image features are extracted at each vertex (a). Since the contour is a cycle graph, circular convolution is applied for feature learning on the contour (b). The blue, yellow and green nodes denote the input features, the kernel of circular convolution, and the output features, respectively. Finally, offsets are regressed at each vertex to deform the contour to match the object boundary (c).

Some recent learning-based segmentation methods [20, 42, 41] also represent objects as contours and try to directly regress the coordinates of contour vertices from an RGB image. Although such methods are fast, most of them do not perform as well as pixel-based methods. Instead, Ling et al. [25] adopt the deformation pipeline of traditional snake algorithms and train a neural network to evolve an initial contour to match the object boundary. Given a contour with image features, it regards the input contour as a graph and uses a graph convolutional network (GCN) to predict vertex-wise offsets between contour points and the target boundary points. It achieves competitive accuracy compared with pixel-based methods while being much faster. However, the method proposed in [25] is designed to help annotation and lacks a complete pipeline for automatic instance segmentation. Moreover, treating the contour as a general graph with a generic GCN does not fully exploit the special topology of a contour.

*The authors from Zhejiang University are affiliated with the State Key Lab of CAD&CG. Corresponding author: Xiaowei Zhou.

In this paper, we propose a learning-based snake algorithm, named deep snake, for real-time instance segmentation. Inspired by previous methods [21, 25], deep snake takes an initial contour as input and deforms it by regressing vertex-wise offsets. Our innovation is introducing the circular convolution for efficient feature learning on a contour, as illustrated in Figure 1. We observe that the contour is a cycle graph that consists of a sequence of vertices connected in a closed cycle. Since every vertex has the same degree equal to two, we can apply the standard 1D convolution on the vertex features. Considering that the contour is periodic, deep snake introduces the circular convolution, which indicates that an aperiodic function (1D kernel) is convolved in the standard way with a periodic function (features defined on the contour). The kernel of circular convolution encodes not only the feature of each vertex but also the relationship among neighboring vertices. In contrast, the generic GCN performs pooling to aggregate information from neighboring vertices. The kernel function in our circular convolution amounts to a learnable aggregation function, which is more expressive and results in better performance than using a generic GCN, as demonstrated by our experimental results in Section 5.2.

Based on deep snake, we develop a pipeline for instance segmentation. Given an initial contour, deep snake can iteratively deform it to match the object boundary and obtain the object shape. The remaining question is how to initialize a contour, whose importance has been demonstrated in classic snake algorithms. Inspired by [32, 29, 45], we propose to generate an octagon formed by object extreme points as the initial contour, which generally encloses the object tightly. Specifically, we integrate deep snake with an object detector. The detected bounding box initializes a diamond contour defined by four center points on the edges. Then, deep snake takes the diamond as input and outputs offsets that point from diamond vertices to object extreme points, which are used to construct an octagon following [45]. Finally, deep snake deforms the octagon contour to match the object boundary.

Our approach exhibits competitive performances on Cityscapes [8], KINS [35], SBD [16] and COCO [24] datasets, while being efficient for real-time instance segmentation, 32.3 fps for 512×512 images on a GTX 1080ti GPU. The following two facts make learning-based snake fast and accurate. First, our approach can deal with errors in the object localization stage and thus allows a light detector. Second, the contour representation has fewer parameters than the pixel-based representation and does not require costly post-processing, e.g., mask upsampling.

In summary, this work has the following contributions:

- We propose a learning-based snake algorithm for real-time instance segmentation and introduce the circular convolution for feature learning on the contour.
- We propose a two-stage pipeline for instance segmentation: initial contour proposal and contour deformation. Both stages can deal with errors in the initial object localization.
- We demonstrate state-of-the-art performances of our approach on Cityscapes, KINS, SBD and COCO datasets. For 512×512 images, our algorithm runs at 32.3 fps, which is efficient for real-time applications.

2. Related work

Pixel-based methods. Most methods [9, 23, 18, 27] perform instance segmentation on the pixel level within a region proposal, which works particularly well with standard CNNs. A representative instantiation is Mask R-CNN [18]. It first detects objects and then uses a mask predictor to segment instances within the proposed boxes. To better exploit the spatial information inside the box, PANet [27] fuses mask predictions from fully-connected layers and convolutional layers. Such proposal-based approaches achieve state-of-the-art performance. One limitation of these methods is that they cannot resolve errors in localization, such as too small or shifted boxes. In contrast, our approach deforms the detected boxes to the object boundaries, so the spatial extension of object shapes will not be limited.

There exist some pixel-based methods [2, 31, 28, 12, 43] that are free of region proposals. In these methods, every pixel produces the auxiliary information, and then a clustering algorithm groups pixels into object instances based on their information. The auxiliary information and grouping algorithms could be various. [2] predicts the boundary-aware energy for each pixel and uses the watershed transform algorithm for grouping. [31] differentiates instances by learning instance-level embeddings. [28, 12] consider the input image as a graph and regress pixel affinities, which are then processed by a graph merge algorithm. Since the mask is composed of dense pixels, the post-clustering algorithms tend to be time-consuming.

Contour-based methods. In these methods, the object shape comprises a sequence of vertices along the object boundary. Traditional snake algorithms [21, 6, 7, 15] first introduced the contour-based representation for image segmentation. They deform an initial contour to the object boundary by optimizing a handcrafted energy with respect to the contour coordinates. To improve the robustness of these methods, [30] proposed to learn the energy function in a data-driven manner. Instead of iteratively optimizing the contour, some recent learning-based methods [20, 42] try to regress the coordinates of contour points from an RGB image, which is much faster. However, their reported accuracy is not on par with state-of-the-art pixel-based methods.

In the field of semi-automatic annotation, [4, 1, 25] have tried to perform the contour labeling using other networks instead of standard CNNs. [4, 1] predict the contour points sequentially using a recurrent neural network. To avoid sequential inference, [25] follows the pipeline of snake algorithms and uses a graph convolutional network to predict vertex-wise offsets for contour deformation. This strategy significantly improves the annotation speed while being as accurate as pixel-based methods. However, [25] lacks a pipeline for instance segmentation and does not fully exploit the special topology of a contour. Instead of treating the contour as a general graph, deep snake leverages the cycle graph topology and introduces the circular convolution for efficient feature learning on a contour.

3. Proposed approach

Inspired by [21, 25], we perform object segmentation by deforming an initial contour to match object boundary. Specifically, deep snake takes a contour as input and predicts per-vertex offsets pointing to the object boundary. Features on contour vertices are extracted from the input image with a CNN backbone. To fully exploit the contour topology, we propose the circular convolution for efficient feature learning on the contour, which facilitates deep snake to learn the deformation. Based on deep snake, we also develop a pipeline for instance segmentation.

3.1. Learning-based snake algorithm

Given an initial contour, traditional snake algorithms treat the coordinates of the vertices as a set of variables and optimize an energy functional with respect to these variables. By designing proper forces at the contour coordinates, the algorithms could drive the contour to the object boundary. However, since the energy functional is typically nonconvex and handcrafted based on low-level image features, the optimization tends to find local optimal solutions.

In contrast, deep snake directly learns to evolve the contour in an end-to-end manner. Given a contour with N vertices $\{\mathbf{x}_i | i = 1, ..., N\}$, we first construct feature vectors for each vertex. The input feature f_i for a vertex $\mathbf{x}_i$ is a concatenation of learning-based features and the vertex coordinate: $[F(\mathbf{x}_i); \mathbf{x}_i]$, where F denotes the feature maps. The feature maps F are obtained by applying a CNN backbone on the input image. The CNN backbone is shared with the detector in our instance segmentation pipeline, which will be discussed later. The image feature $F(\mathbf{x}_i)$ is computed using the bilinear interpolation at the vertex coordinate $\mathbf{x}_i$. The appended vertex coordinate is used to encode the spatial relationship among contour vertices. Since the deformation should not be affected by the translation of the contour in the image, we subtract each dimension of $\mathbf{x}_i$ by the minimum value over all vertices.

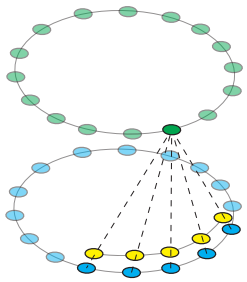

Figure 2. **Circular Convolution.** The blue nodes are the input features defined on a contour, the yellow nodes represent the kernel function, and the green nodes are the output features. The highlighted green node is the inner product between the kernel function and the highlighted blue nodes, which is the same as the standard convolution. The output features of circular convolution have the same length as the input features.

Given the input features defined on a contour, deep snake introduces the circular convolution for the feature learning, as illustrated in Figure 2. In general, the features of contour vertices can be treated as a 1-D discrete signal $f : \mathbb{Z} \rightarrow \mathbb{R}^D$ and processed by the standard convolution. But this breaks the topology of the contour. Therefore, we extend f to be a periodic signal defined as:

$$(f_N)_i \triangleq \sum_{j=-\infty}^{\infty} f_{i-jN}, \tag{1}$$

and propose to encode the periodic features by the circular convolution defined as:

$$(f_N * k)_i = \sum_{j=-r}^{r} (f_N)_{i+j} k_j, \tag{2}$$

where $k : [-r, r] \rightarrow \mathbb{R}^D$ is a learnable kernel function and the operator $*$ is the standard convolution.

Similar to the standard convolution, we can construct a network layer based on the circular convolution for feature learning, which is easy to be integrated into a modern network architecture. After the feature learning, deep snake applies three 1×1 convolution layers to the output features for each vertex and predicts vertex-wise offsets between contour points and the target points, which are used to deform the contour. In all experiments, the kernel size of circular convolution is fixed to be nine.

As discussed in the introduction, the proposed circular convolution better exploits the circular structure of the contour than the generic graph convolution. We will show the experimental comparison in Section 5.2. An alternative method is to use standard CNNs to regress a pixel-wise vector field from the input image to guide the evolution of the initial contour [37, 33, 40]. We argue that an important advantage of deep snake over the standard CNNs is the object-level structured prediction, i.e., the offset prediction at a vertex depends on other vertices of the same contour. Therefore, deep snake will predict a more reasonable offset for a vertex located far from the object. Standard CNNs may have difficulty in this case, as the regressed vector field may drive this vertex to another object which is closer.

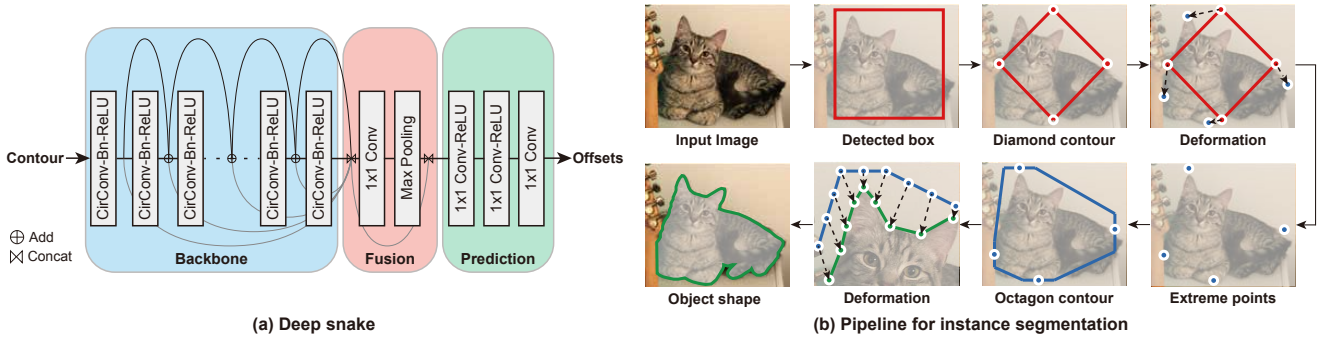

Figure 3. **Proposed contour-based model for instance segmentation.** (a) Deep snake consists of three parts: a backbone, a fusion block, and a prediction head. It takes a contour as input and outputs vertex-wise offsets to deform the contour. (b) Based on deep snake, we propose a two-stage pipeline for instance segmentation: initial contour proposal and contour deformation. The box proposed by the detector gives a diamond contour, whose four vertices are then shifted to object extreme points by deep snake. An octagon is constructed based on the extreme points. Taking the octagon as the initial contour, deep snake iteratively deforms it to match the object boundary.

Network architecture. Figure 3(a) shows the detailed schematic. Following ideas from [34, 39, 22], deep snake consists of three parts: a backbone, a fusion block, and a prediction head. The backbone is comprised of 8 "CirConv-Bn-ReLU" layers and uses residual skip connections for all layers, where "CirConv" means circular convolution. The fusion block aims to fuse the information across all contour points at multiple scales. It concatenates features from all layers in the backbone and forwards them through a 1×1 convolution layer followed by max pooling. The fused feature is then concatenated with the feature of each vertex. The prediction head applies three 1×1 convolution layers to the vertex features and output vertex-wise offsets.

3.2. Deep snake for instance segmentation

Figure 3(b) overviews the proposed pipeline for instance segmentation. We combine deep snake with an object detector. The detector first produces object bounding boxes that are used to construct diamond contours. Then deep snake shifts the diamond vertices to object extreme points, which are used to construct octagon contours. Finally, deep snake takes octagons as initial contours and performs iterative contour deformation to obtain the object shape.

Initial contour proposal. Most active contour models require precise initial contours. Since the octagon proposed in [45] tightly encloses the object, we choose it as the initial contour, as shown in Figure 3(b). This octagon is formed by four extreme points, which are top, leftmost, bottom, rightmost pixels in an object, respectively, denoted by $\{\mathbf{x}_i^{ex}|i=1,2,3,4\}$. Given a detected object box, we extract four center points at the top, left, bottom, right box edges, denoted by $\{\mathbf{x}_i^{bb}|i=1,2,3,4\}$, and then connect them to get a diamond contour. Deep snake takes this contour as input and outputs four offsets that point from each vertex $\mathbf{x}_i^{bb}$ to the extreme point $\mathbf{x}_i^{ex}$, namely $\mathbf{x}_i^{ex}-\mathbf{x}_i^{bb}$. In practice, to consider more context information, the diamond contour is uniformly upsampled to 40 points, and deep snake correspondingly outputs 40 offsets. The loss function only supervises the offsets at $\mathbf{x}_i^{bb}$.

We construct the octagon by generating four line segments based on extreme points and connecting their endpoints. Specifically, the four extreme points define a new bounding box. From each extreme point, a line is extended along the corresponding box edge in both directions by 1/4 of the edge length and truncated if it meets the box corner. Then, the endpoints of the four line segments are connected to form the octagon.

Contour deformation. We first uniformly sample N points along the octagon contour starting from the top extreme points $\mathbf{x}_1^{ex}$. Similarly, the ground-truth contour is generated by uniformly sampling N vertices along the object boundary and defining the first vertex as the one nearest to $\mathbf{x}_1^{ex}$. Deep snake takes the initial contour as input and outputs N offsets that point from each vertex to the target boundary point. We set N as 128 in all experiments, which can uniformly cover most object shapes.

However, regressing the offsets in one pass is challenging, especially for vertices far away from the object. Inspired by [21, 25, 38], we deal with this problem in an iterative optimization fashion. Specifically, our approach first predicts N offsets based on the current contour and then deforms this contour by vertex-wise adding the offsets to its vertex coordinates. The deformed contour can be used for the next iteration. In experiments, the number of inference iteration is set as 3 unless otherwise stated.

Note that the contour is an alternative representation for the spatial extension of an object. By deforming the initial contour to match the object boundary, deep snake could address the localization errors from the detector.

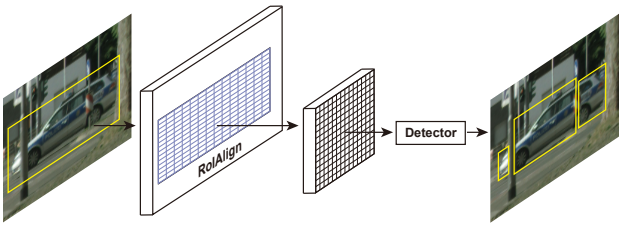

Figure 4. **Multi-component detection.** Given an object box, we perform RoIAlign to obtain the feature map and use a detector to detect the component boxes.

Multi-component detection. Some objects are split into several components due to occlusions, as shown in Figure 4. However, a contour can only outline one component. To overcome this problem, we propose to use another detector to find the object components within the object box. Figure 4 shows the basic idea. Specifically, using the detected box, our approach performs RoIAlign [18] to extract a feature map and adds a detector branch on the feature map to produce the component boxes. For the detected components, we use deep snake to segment each of them and then merge the segmentation results.

4. Implementation details

Training strategy. For the training of deep snake, we use the smooth ℓ_1 loss proposed in [14] to learn the two deformation processes. The loss function for extreme point prediction is defined as

$$L_{ex} = \frac{1}{4}\sum_{i=1}^{4} \ell_1(\tilde{\mathbf{x}}_i^{ex} - \mathbf{x}_i^{ex}), \tag{3}$$

where $\tilde{\mathbf{x}}_i^{ex}$ is the predicted extreme point. And the loss function for iterative contour deformation is defined as

$$L_{iter} = \frac{1}{N}\sum_{i=1}^{N} \ell_1(\tilde{\mathbf{x}}_i - \mathbf{x}_i^{gt}), \tag{4}$$

where $\tilde{\mathbf{x}}_i$ is the deformed contour point and $\mathbf{x}_i^{gt}$ is the ground-truth boundary point. For the detection part, we adopt the same loss function as the original detection model. The training details change with datasets, which will be described in Section 5.3.

Detector. We adopt CenterNet [44] as the detector for all experiments. CenterNet reformulates the detection task as a keypoint detection problem and achieves an impressive trade-off between speed and accuracy. For the object box detector, we adopt the same setting as [44], which outputs class-specific boxes. For the component box detector, a class-agnostic CenterNet is adopted. Specifically, given an $H \times W \times C$ feature map, the class-agnostic CenterNet outputs an $H \times W \times 1$ tensor representing the component center and an $H \times W \times 2$ tensor representing the box size.

5. Experiments

5.1. Datasets and Metrics

Cityscapes [8] contains $2,975$ training, 500 validation and $1,525$ testing images with high quality annotations. Besides, it has 20k images with coarse annotations. The performance is evaluated in terms of the average precision (AP) metric averaged over eight semantic classes of the dataset.

KINS [35] was created by additionally annotating Kitti [13] dataset with instance-level semantic annotation. This dataset is used for amodal instance segmentation, which aims to recover complete instance shapes even under occlusion. KINS consists of $7,474$ training images and $7,517$ testing images. Following its setting, we evaluate our approach on seven object categories in terms of the AP metric.

SBD [16] re-annotates $11,355$ images from the PASCAL VOC [10] dataset with instance-level boundaries. The reason that we don't evaluate on PASCAL VOC is that its annotations contain holes, which is not suitable for contour-based methods. SBD is split into $5,623$ training images and $5,732$ testing images. We report our results in terms of 2010 VOC AP_{vol} [17], AP_{50}, AP_{70} metrics. AP_{vol} is the average of AP with nine thresholds from 0.1 to 0.9.

COCO [24] is one of the most challenging datasets for instance segmentation. It consists of 115k training , 5k validation and 20k testing images. We report our results in terms of the AP metric.

5.2. Ablation studies

We conduct ablation studies on the SBD dataset as it has 20 semantic categories which could fully evaluate the ability to handle various object shapes. The three proposed components are evaluated, including our network architecture, initial contour proposal, and circular convolution. In these experiments, the detector and deep snake are trained end-to-end for 160 epochs with multi-scale data augmentation. The learning rate starts from $1e^{-4}$ and decays by half at 80 and 120 epochs.

Table 1 summarizes the results of ablation studies. The row "Baseline" lists the result of a direct combination of Curve-gcn [25] with CenterNet [44]. Specifically, the detector produces object boxes, which gives ellipses around objects. Then ellipses are deformed towards object boundaries through Graph-ResNet. Note that, this baseline method represents the contour as a graph and uses a graph convolution network for contour deformation.

To validate the advantages of our network, the model in the second row keeps the convolution operator as graph convolution and replaces Graph-ResNet with our proposed architecture, which yields 1.4 AP_{vol} improvement. The main

	AP_{vol}	AP_{50}	AP_{70}
Baseline	50.9	58.8	43.5
+ Architecture	52.3	59.7	46.0
+ Initial proposal	53.6	61.1	47.6
+ Circular convolution	**54.4**	**62.1**	**48.3**

Table 1. **Ablation studies on SBD val set .** The baseline is a direct combination of Curve-gcn [25] and CenterNet [44]. The second model reserves the graph convolution and replaces the network architecture with our proposed one, which yields 1.4 AP_{vol} improvement. Then we add the initial contour proposal before contour deformation, which improves AP_{vol} by 1.3. The fourth row shows that replacing graph convolution with circular convolution further yields 0.8 AP_{vol} improvement.

	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5
Graph conv	50.2	51.5	53.6	52.2	51.6
Circular conv	50.6	54.2	**54.4**	54.0	53.2

Table 2. **Results of models with different convolution operators and different iterations** on SBD in terms of the AP_{vol} metric. Circular convolution outperforms graph convolution across all inference iterations. Furthermore, circular convolution with two iterations outperforms graph convolution with three iterations by 0.6 AP, indicating a stronger deforming ability. We also find that adding more iterations does not necessarily improve the performance, which shows that it might be harder to train the network with more iterations.

difference between the two networks is that our architecture appends a global fusion block before the prediction head.

When exploring the influence of the contour initialization, we add the initial contour proposal before the contour deformation. Instead of directly using the ellipse, the proposal step generates an octagon initialization by predicting four object extreme points, which not only compensates for the detection errors but also encloses the object more tightly. The comparison between the second and the third row shows a 1.3 improvement in terms of AP_{vol}.

Finally, the graph convolution is replaced with the circular convolution, which achieves 0.8 AP_{vol} improvement. To fully validate the importance of circular convolution, we further compare models with different convolution operators and different inference iterations, as shown in table 2. Circular convolution outperforms graph convolution across all inference iterations. Circular convolution with two iterations outperforms graph convolution with three iterations by 0.6 AP_{vol}. Figure 5 shows qualitative results of graph and circular convolution on SBD, where circular convolution gives a sharper boundary. Both the quantitative and qualitative results indicate that models with the circular convolution have a stronger ability to deform contours.

5.3. Comparison with the state-of-the-art methods

Performance on Cityscapes. Since fragmented instances are very common in Cityscapes, the proposed multi-component detection strategy is adopted. Our network is trained with multi-scale data augmentation and tested at a single resolution of 1216×2432. No testing tricks are used. The detector is first trained alone for 140 epochs, and the learning rate starts from $1e^{-4}$ and drops by half at 80, 120 epochs. Then the detection and snake branches are trained end-to-end for 200 epochs, and the learning rate starts from $1e^{-4}$ and drops by half at 80, 120, 150 epochs. We choose a model that performs best on the validation set.

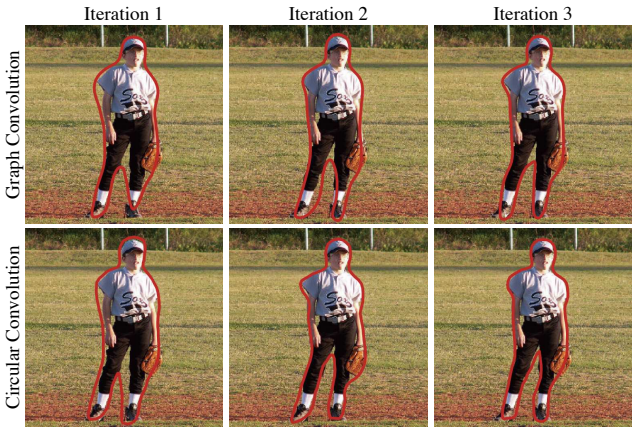

Figure 5. **Comparison between graph convolution (top) and circular convolution (bottom) on SBD.** The result of circular convolution with two iterations is visually better than that of graph convolution with three iterations.

Table 3 compares our results with other state-of-the-art methods on the Cityscapes validation and test sets. All methods are tested without tricks. Using only the fine annotations, our approach achieves state-of-the-art performances on both validation and test sets. We outperform PANet by 0.9 AP on the validation set and 1.3 AP_{50} on the test set. Our approach achieves 28.2 AP on the test set when the strategy of handling fragmented instances is not adopted. Visual results are shown in Figure 6.

Performance on KINS. The KINS dataset is for amodal instance segmentation, where objects are all annotated as single-component, so the multi-component detection strategy is not adopted. We train the detector and snake end-to-end for 150 epochs. The learning rate starts from $1e^{-4}$ and decays with 0.5 and 0.1 at 80 and 120 epochs, respectively. We perform multi-scale training and test the model at a single resolution of 768×2496.

Table 4 shows the comparison with [9, 23, 11, 18, 27] on the KINS dataset in terms of the AP metric. Our approach achieves the best performance across all methods. We find that the snake branch can improve the detection performance. When CenterNet is trained alone, it obtains 30.5 AP on detection. When trained with the snake branch, its performance improves by 2.3 AP. For an image resolution of 768×2496 on the KINS dataset, our approach runs at 7.6 fps on a 1080 Ti GPU. Figure 6 shows some qualitative results on KINS.

Figure 6. **Qualitative results on Cityscapes test and KINS test sets.** The first two rows show the results on Cityscapes, and the last row lists the results on KINS. Note that the results on KINS are for amodal instance segmentation.

	training data	fps	AP [val]	AP	AP_{50}	person	rider	car	truck	bus	train	mcycle	bicycle
SGN [26]	fine + coarse	0.6	29.2	25.0	44.9	21.8	20.1	39.4	24.8	33.2	30.8	17.7	12.4
PolygonRNN++ [1]	fine	-	-	25.5	45.5	29.4	21.8	48.3	21.1	32.3	23.7	13.6	13.6
Mask R-CNN [18]	fine	2.2	31.5	26.2	49.9	30.5	23.7	46.9	22.8	32.2	18.6	19.1	16.0
GMIS [28]	fine + coarse	-	-	27.6	49.6	29.3	24.1	42.7	25.4	37.2	**32.9**	17.6	11.9
Spatial [31]	fine	11	-	27.6	50.9	34.5	26.1	52.4	21.7	31.2	16.4	20.1	18.9
PANet [27]	fine	<1	36.5	**31.8**	57.1	36.8	**30.4**	54.8	27.0	36.3	25.5	**22.6**	**20.8**
Deep snake	fine	4.6	**37.4**	31.7	**58.4**	**37.2**	27.0	**56.0**	**29.5**	**40.5**	28.2	19.0	16.4

Table 3. **Results on Cityscapes val ("AP [val]" column) and test (remaining columns) sets.** Our approach achieves the state-of-the-art performance, which outperforms PANet [27] by 0.9 AP on the val set and 1.3 AP_{50} on the test set. In terms of the inference speed, our approach is approximately five times faster than PANet. The timing results of other methods were obtained from [31].

	detection	amodal seg	inmodal seg
MNC [9]	20.9	18.5	16.1
FCIS [23]	25.6	23.5	20.8
ORCNN [11]	30.9	29.0	26.4
Mask R-CNN [18]	31.1	29.2	×
Mask R-CNN [18]	31.3	29.3	26.6
PANet [27]	32.3	30.4	27.6
Deep snake	**32.8**	**31.3**	×

Table 4. **Results on KINS test set in terms of the AP metric.** The amodal bounding box is used as the ground truth in the detection task. × means no such output in the corresponding method.

Performance on SBD. Since annotations of objects on SBD are mostly single-component, the multi-component detection strategy is not adopted. For fragmented instances, our approach detects their components separately instead of detecting the whole object. We train the detection and snake branches end-to-end for 150 epochs with multi-scale data augmentation. The learning rate starts from $1e^{-4}$ and drops by half at 80 and 120 epochs. The network is tested at a single scale of 512×512.

	AP_{vol}	AP_{50}	AP_{70}
STS [20]	29.0	30.0	6.5
ESE-50 [42]	32.6	39.1	10.5
ESE-20 [42]	35.3	40.7	12.1
Deep snake	**54.4**	**62.1**	**48.3**

Table 5. **Results on SBD val set.** Our approach outperforms other contour-based methods by a large margin. The improvement increases with the IoU threshold: 21.4 in AP_{50} and 36.2 in AP_{70}.

In Table 5, we compare with other contour-based methods [20, 42] on the SBD dataset in terms of the VOC AP metrics. [20, 42] predict the object contours by regressing shape vectors. STS [20] defines the object contour as a radial vector, and ESE [42] approximates object contour with the Chebyshev polynomial. We outperform these methods by a large margin of at least 19.1 AP_{vol}. Note that, our approach yields 21.4 AP_{50} and 36.2 AP_{70} improvements, demonstrating that the improvement increases as the IoU threshold gets smaller. This indicates that our method outlines object boundaries more precisely. For 512×512 im-

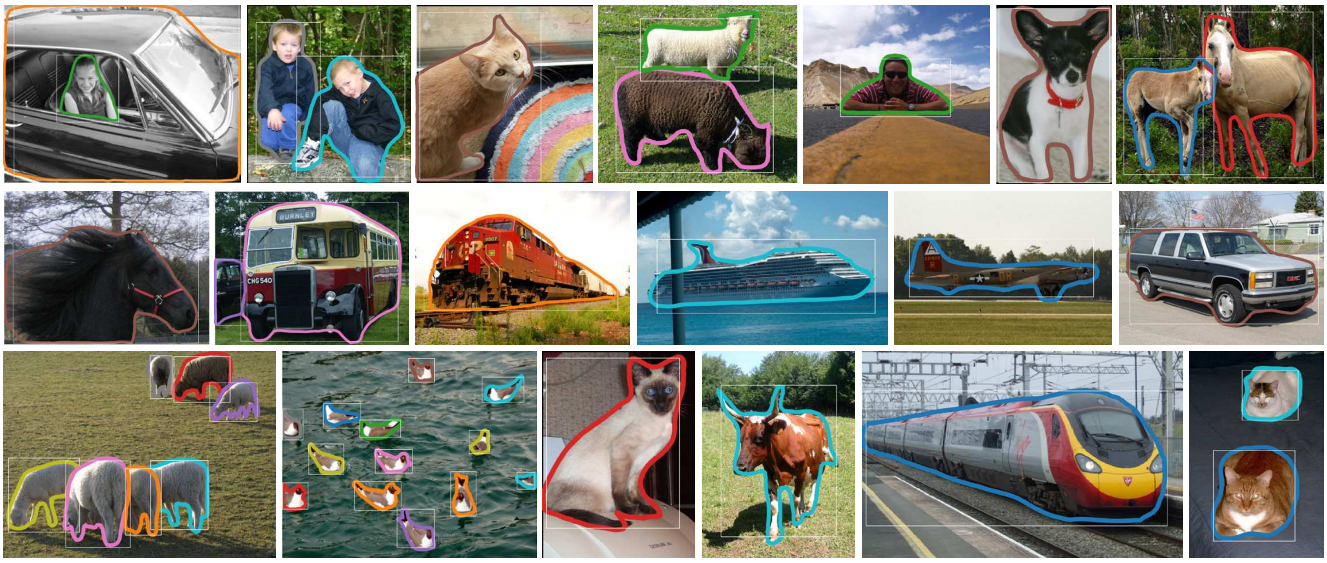

Figure 7. **Qualitative results on SBD val set.** Our approach handles errors in object localization in most cases. For example, in the first image, although the detected box doesn't fully enclose the car, our approach recovers the complete car shape. Zoom in for details.

	YOLACT [3]	ESE [42]	OURS
val (segm AP)	29.9	21.6	**30.5**
test-dev (segm AP)	29.8	-	**30.3**

Table 6. **Comparison with other real-time methods** on COCO.

ages on the SBD dataset, our approach runs at 32.3 fps on a 1080 Ti. Some qualitative results are illustrated in Figure 7.

Performance on COCO. Similar to the experiment on SBD, the multi-component detection strategy is not adopted. The network is trained with multi-scale data augmentation and tested at the original image resolution without tricks (e.g., flip augmentation). The detection and snake branches are trained end-to-end for 160 epochs, where the detector is initialized with the pretrained model released by [44]. The learning rate starts from $1e^{-4}$ and drops by half at 80 and 120 epochs. We choose a model that performs best on the validation set. Table 6 compares our method with other real-time methods. Our method achieves 30.3 segm AP and 33.2 bbox AP on COCO test-dev set with 27.2 fps.

5.4. Running time

Table 7 compares our approach with other methods [9, 23, 18, 20, 42] in terms of running time on the PASCAL VOC dataset. Since the SBD dataset shares images with PASCAL VOC, the running time on the SBD dataset is technically the same as the one on PASCAL VOC. We obtain the running time of other methods from [42].

For 512×512 images on the SBD dataset, our algorithm runs at 32.3 fps on a desktop with an Intel i7 3.7GHz and a GTX 1080 Ti GPU, which is efficient for real-time instance segmentation. Specifically, CenterNet takes 18.4 ms, the initial contour proposal takes 3.1 ms, and each iteration of contour deformation takes 3.3 ms. Since our approach outputs the object boundary, no post-processing like upsampling is required. If the multi-component detection strategy is adopted, the detector additionally takes 3.6 ms.

method	MNC	FCIS	MS	STS	ESE	OURS
time (ms)	360	160	180	27	26	31
fps	2.8	6.3	5.6	37.0	38.5	32.3

Table 7. **Running time on the PASCAL VOC dataset.** "MS" represents Mask R-CNN [18] and "OURS" represents our approach. The last three methods are contour-based methods.

6. Conclusion

We proposed a learning-based snake algorithm for real-time instance segmentation, which introduces the circular convolution for efficient feature learning on the contour and regresses vertex-wise offsets for the contour deformation. Based on deep snake, we developed a two-stage pipeline for instance segmentation: initial contour proposal and contour deformation. We showed that this pipeline gained a superior performance than direct regression of the coordinates of the object boundary points. To overcome the limitation of the contour representation that it can only outline one connected component, we proposed the multi-component detection strategy and demonstrated the effectiveness of this strategy on Cityscapes. The proposed model achieved competitive results on the Cityscapes, Kins, Sbd and COCO datasets with a real-time performance.

Acknowledgements: The authors would like to acknowledge support from NSFC (No. 61806176) and Fundamental Research Funds for the Central Universities (2019QNA5022).

References

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, 2018. 3, 7

[2] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017. 2

[3] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: real-time instance segmentation. In *ICCV*, 2019. 8

[4] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017. 3

[5] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, 2019. 1

[6] Laurent D Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991. 1, 2

[7] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *CVIU*, 61(1):38–59, 1995. 1, 2

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 5

[9] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016. 2, 6, 7, 8

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 5

[11] Patrick Follmann, Rebecca Kö Nig, Philipp Hä Rtinger, Michael Klostermann, and Tobias Bö Ttger. Learning to see the invisible: End-to-end trainable amodal instance segmentation. In *WACV*, 2019. 6, 7

[12] 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. 2

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 32(11):1231–1237, 2013. 5

[14] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 5

[15] Steve R Gunn and Mark S Nixon. A robust snake implementation; a dual active contour. *PAMI*, 19(1):63–68, 1997. 1, 2

[16] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 2, 5

[17] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 5

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 2, 5, 6, 7, 8

[19] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR*, 2019. 1

[20] Saumya Jetley, Michael Sapienza, Stuart Golodetz, and Philip HS Torr. Straight to shapes: Real-time detection of encoded shapes. In *CVPR*, 2017. 1, 2, 7, 8

[21] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988. 1, 2, 3, 4

[22] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Can gcns go as deep as cnns? In *ICCV*, 2019. 4

[23] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *CVPR*, 2017. 2, 6, 7, 8

[24] 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. 2, 5

[25] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, 2019. 1, 2, 3, 4, 5, 6

[26] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *ICCV*, 2017. 7

[27] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 1, 2, 6, 7

[28] 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. 2, 7

[29] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018. 2

[30] 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. 2

[31] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *CVPR*, 2019. 2, 7

[32] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017. 2

[33] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019. 3

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 4

[35] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. In *CVPR*, 2019. 2, 5

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1

[37] Christian Rupprecht, Elizabeth Huaroc, Maximilian Baust, and Nassir Navab. Deep active contours. *arXiv preprint arXiv:1607.05074*, 2016. 3

[38] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 4

[39] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 2018. 4

[40] Zian Wang, David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Object instance annotation with deep extreme level set evolution. In *CVPR*, 2019. 3

[41] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *CVPR*, 2020. 1

[42] Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *ICCV*, 2019. 1, 2, 7, 8

[43] Ze Yang, Yinghao Xu, Han Xue, Zheng Zhang, Raquel Urtasun, Liwei Wang, Stephen Lin, and Han Hu. Dense reppoints: Representing visual objects with dense point sets. *arXiv preprint arXiv:1912.11473*, 2019. 2

[44] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 5, 6, 8

[45] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019. 2, 4