Tsnake: A Time-Embedded Recurrent Contour-Based Instance Segmentation Model

Chen-Jui Hsu[1], Jian-Jiun Ding [2*], and Chun-Jen Shih[3]
[1] National Taiwan University, Taiwan
E-mail: r11942085@ntu.edu.tw Tel: +886-233663662
[2*] National Taiwan University, Taiwan
E-mail: jjding@ntu.edu.tw Tel: +886-233669652
[3] National Taiwan University, Taiwan
E-mail: f10942045@ntu.edu.tw Tel: +886-233663662

***Abstract* — Instance segmentation is a critical task in computer vision. It is widely applied in autonomous driving and medical image analysis. Contour-based methods have been studied for their elegant structures, compact representation, and low inference time. In existing work, contour-based methods apply iterative reformation to fit the object boundary, however, the modules are independent in each loop, which enlarges the model sizes. In this paper, we introduce the TSnake. It is a novel time-embedded driven recurrent contour-based instance segmentation model. At each iteration, the refinement module adjusts the contour with the help of semantic-rich features and the time step. Moreover, a dilated module is applied to well adopt the local contextual information. We also modify the loss function to well balance the trade-off between over-smoothing and over-sensitivity of the contour. The proposed TSnake method surpasses the contour-based state-of-the-art instance segmentation and can be performed in real time.**

I. Introduction

Instance segmentation, which aims to detect and delineate each distinct object instance in an image, is a fundamental yet challenging task in computer vision. It plays a crucial role in various applications, including autonomous driving robotics, medical imaging, and augmented reality.

Most approaches to instance segmentation can be broadly categorized into two main types: pixel-based and contour-based. Pixel-based methods predict the masks of the objects directly [1-5]. Some of them perform a two-stage pipeline, in which the first stage detects the bounding boxes of objects and the latter predicts the masks, to achieve higher accuracy. These designs cause intensive computation and slow inference time. Meanwhile, some directly predict masks from the backbone to reach real-time speed.

As for contour-based models [6-11], they predict several points or coefficients and form them into a polygon to approach the boundary of the objects. For example, Xie *et al.* [6] proposed PolarMask which predicts 36 segments started from the object center uniformly to generate a polygon, and Xu *et al.* [7] applied Chebyshev polynomials to fit the shape. Contour-based methods only predict distinct points or coefficients, thus the computation costs are usually less than pixel-based methods.

Inspired by the active contour model [11], in [8-10], they proposed iterative refinement pipelines to perform instance segmentation. They all apply a module with several layers of circular convolution, or "snake", to refine the local segment to approach the boundary of the object. These recurrent contour-based models treat object segmentation as a regression problem. However, existing contour-based methods still face several challenges, including over-smoothed or over-sensitive contours, redundant information during iteration and limited receptive fields, etc.

In this paper, we introduce TSnake, a recurrent contour-based instance segmentation model to address these limitations. Our contributions includes:

1. A time-embedding controlled snake module that enables iterative refinement without additional information while maintaining weight sharing is proposed.
2. A dilated module that expands the receptive field is proposed. It can well apply local contextual information.
3. A weighted dynamic matching loss is proposed. It well balances the trade-off between over-smoothing and over-sensitivity of the contour.

Our experiments on the SBD and Cityscapes datasets demonstrate that the proposed TSnake achieves state-of-the-art performance among contour-based instance segmentation methods while maintaining real-time efficiency.

II. Related Work

A. Pixel-based Instance Segmentation

Pixel-based instance segmentation methods can be classified into: two-stage methods and one-stage methods. Two-stage methods basically follow the pipeline of the Mask RCNN [1]. In the first stage, they predict several bounding boxes that probably contain objects. The second stage is for classification. In [2, 12], they applied cascade prediction series to improve masks iteratively. Shen *et al.* [3] applied the DCT to reduce the loss from resizing.

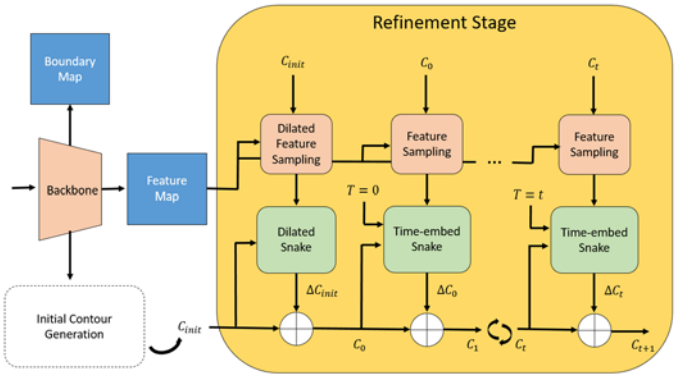

Fig. 1 The overall structure of TSnake. For an image passed to the model, initial contour generation will predict C_{init} and pass it to the dilated snake to generate C_0. C_0 will be deformed for T times by the weight sharing time-embed snake.

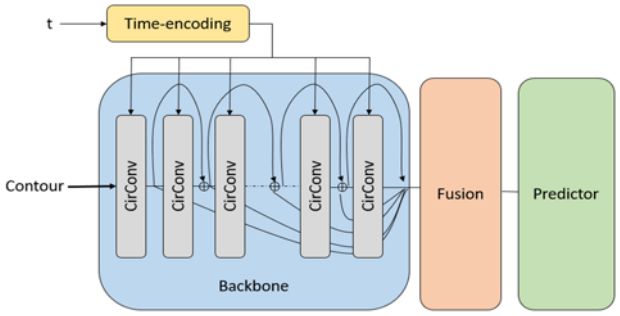

Fig. 2 Structure of the time-embed snake. The sequence t will be encoded by sinusoid functions and added to all circular convolution in the snake backbone.

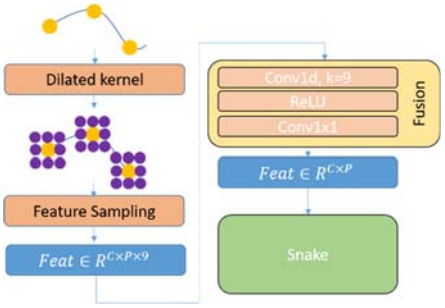

Fig. 3 Pipeline of the dilated snake. The dilated kernel fetches the samples around the points in the coordinate system of the origin image.

By contrast, one-stage methods directly predict masks from the backbone, making them usually have low inference time. Bolya *et al.* [4] predicted plenty of protomasks to present each mask of an object. Wang *et al.* [5] separated the origin image into N^2 grids and predicted a mask for each grid. These methods generally offer faster inference times but may struggle with fine-grained details and high memory usage.

B. Contour-based Instance Segmentation

Contour-based methods focus on object boundaries rather than full masks. They transform instance segmentation into a regression problem, predicting points or coefficients to form polygons. Xu *et al.* [7] applied Chebyshev polynomials and Wang *et al.* [13] predicted the coefficients of the Fourier series to generate results. In another way, directly predicting the coordinates of the points makes the predicted contour much fitter. Xie *et al.* [6] divided 360 degrees into 36 pieces uniformly to predict surrounding points. Peng *et al.* [8] proposed Deep Snake to iteratively refine the contour. DANCE [14] followed Deep Snake and added an attention mechanism to fetch more edge information.

Also, Zhang *et al.* [9] modified initial contour generation of [8] to learning-based to achieve better performance, and Feng *et al.* [10] decreased the model weights by an RNN-like weight-sharing refinement stage. However, most of them do not share the weights for iteration. Also, the contours of the above recurrent instance segmentation models are sometimes too over-smooth to enclose the extreme points, or too sensitive to noise which makes the contour jagged.

III. Proposed Method

A. Time-Embedded Snake

We address two issues from previous work. The first one is the contours are over-smooth or over-sensitive, and the second is the weight-sharing module is needed while the information from the previous state is unnecessary.

We propose TSnake, a time-embedding controlled weight-sharing recurrent refinement model that makes a balance between too smooth and too sharp. The overall pipeline is shown in Fig. 1. Our TSnake consists of three main components: a backbone network for extraction, an initial contour generation module inherent from [9], and a refinement stage which is the core innovation of Tsnake. When the image passes into the backbone, the initial contour generation produces an initial contour based on the backbone features, and the contour will be refined to a fit boundary at the refinement stage. Below we will introduce the components of Tsnake and the proposed loss.

Though a weight-sharing scheme has been proposed in [10] in the GRU style, we assume that the features from the previous iteration are not necessary. However, the model still needs some information to tell it which iteration is to do corresponding refinement. Inspired by the diffusion model [15], the diffusion model is a generative model that denoise the Gaussian noise to a clear image with several loops. At each loop the model denoise different levels of noise. The time encoding is added to tell it which iteration is now. We treat the ground truth contour as a clear image, and the initial contour will be like a noisy image. We plugin time embedding into the snake module at the circular convolution layers, which is shown in Fig. 2. We adopt sinusoid functions for time encoding as the same as [15]. The contour deformation can be written as in (1):

$$x^{(t+1)} = snake_T\left(\left[F\left(x^{(t)}\right);\bar{x}^{(t)}\right],t_{enc}\left(t\right)\right)+x^{(t)} \qquad (1)$$

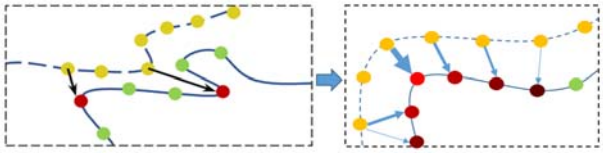

Fig. 4 The representation of the weighted DML. The yellow points are predicted points, red ones are keypoint ground truths, and green ones are the normal ground truth. The left subfigure is the original DML [9] and right one is the proposed weighted DML. The origin DML only attracts one point to the keypoint, which makes the contour sensitive. Our approach builds the corner with much more points to reduce the sensitivity.

where $x^{(t)}$ is the contour coordinates at the t^{th} interation, F is the features from the backbone with 64 channels, and $\bar{x}$ is the translated coordinates that subtract the min value of each dimension. The features from the backbone and the coordinates will then be concatenated together and fed to the module. $snake_T$ is the time-embedded module. $t_{enc}()$ is a function to encode the time step. The iteration loop is set to 8 in our setting.

B. Dilated Snake

We have observed that if the initial contour located at a relatively smooth area, points will be easily misled to the wrong side. Intuitively, we take more features from the neighborhood for a point to decide where is the correct side. In order to not increasing the calculation too much, we employ dilated convolution [16] with kernel size 3. The structure is shown in Fig. 3. Note that the dilated points are in the coordinate system of the origin image. By the dilated kernel, each point will get 9 feature vectors. We fuse these 9 vectors to one vector by Conv1d and pass it to the original snake module. Dilated snake is not only used for deformation with nearby but also to increase the receptive field so that 8N more features can be updated and propagated back to the backbone when training.

C. Weighted DML

In previous works, the contours are usually over-smooth, but E2EC [9] is the one that makes the contour jagged. This is caused by the dynamic matching loss (DML) [9]. The jagged problem occurred because DML only attracts one point to the keypoint. Thus, we assign the adjacent points of keypoints as new keypoints but with different weights. Those new keypoints can attract predicted points as well but with weak forces. The weights are reweighted with Gaussian distribution, where the original keypoints are with the weight 1. The Gaussian kernel of distribution is set as:

$$\exp((x-\text{int}(\mu/2))^2/2\sigma^2). \tag{2}$$

Eq. (2) is sampled in the discrete form and μ and σ are set as 7 and 0.5, respectively. The loss of our weighted DML is:

$$x_i^* = arg\min_x \left\| pred_i^{in} - gt_x^{ipt} \right\|_2 \tag{3}$$

$$L_1 = \frac{1}{N}\sum_{i=1}^{N} SL_1\left(pred_i^{out}, gt_{x_i^*}^{ipt}\right) \tag{4}$$

$$y_i^* = arg\min_y \left\| pred_y^{in} - gt_i \right\|_2 \tag{5}$$

$$L_2 = \frac{1}{sum(K)}\sum_{i=1}^{N} K[i]\times SL_1\left(pred_{y_i^*}^{out}, gt_i\right) \tag{6}$$

$$L_{WDML}\left(pred^{in}, pred^{out}, gt\right) = \frac{L_1+L_2}{2} \tag{7}$$

where SL_1 means the smoothed L_1 loss, *pred* and *gt* means the predicted result and the ground truth, respectively. The part that attract predicted points to the nearest contour remains the same in [9], which is shown in (3) and (4).

D. Loss Function

The loss function is a combination of detection, initial contour, and refinement. Also, a boundary map is adopted to predict to enhance the performance. The losses are shown as follows:

$$L = L_{det} + L_{bd} + L_{init} + L_{refine}, \tag{8}$$

$$\begin{aligned} L_{refine} = &\sum_{k=0}^{4}\lambda^{K-k}\times SL_1\left(x^{(k)}, x\right) \\ &+\sum_{k=5}^{8}\lambda^{K-k}\times wdml\left(x^{(k-1)}, x^{(k)}, x\right) + \sum_{k=0}^{8}\lambda^{K-k}\times shape\left(x^{(k)}, x\right) \end{aligned} \tag{9}$$

where L_{det} is the detection loss, L_{bd} is the boundary loss that applies focal loss [17], L_{init} is the initial contour loss applying the smooth-L_1 loss, λ is a constant less than 1, which we set as 0.8 in experiments, and *wdml* means the weighted DML. L_{refine} is the refinement loss that consists of 3 parts. Also, contours at each iteration are applied to shape loss in [10]. The first four iterations and the contour from the dilated snake are applied smooth-L_1 loss and our weighted DML is used for the latter four iterations.

IV. Experiments and Results

A. Datasets and Metrics

Datasets. The SBD and Cityscapes datasets [18] were used in experiments. The SBD dataset has 20 object classes and 11355 images in total from the PASCAL VOC [19] dataset and is split into a training set for 5623 images and a validation set for 5732 images. Cityscapes has 8 classes that usually appear on the road, and it contains 2975 training images and 500 validation images.

Metrics. We evaluate the mask quality by standard AP. For SBD, we report the performance based on metrics 2010 VOC AP_{vol}, AP_{50}, and AP_{70}. AP_{vol} is the average of AP with nine thresholds from 0.1 to 0.9. For Cityscapes, the results are evaluated in terms of AP metric averaged by 8 categories.

TABLE I: THE RESULT ON THE SBD DATASET WHERE * MEANS THAT MOSAIC AUGMENTATION IS APPLIED.

Name	AP_{vol}	AP_{50}	AP_{70}
ESE-20 [7]	35.3	40.7	12.1
Deep Snake [8]	54.4	62.1	48.3
DANCE [14]	56.2	63.6	50.4
E2EC [9]	59.2	65.8	54.5
TSnake	60.2	66.8	55.5
TSnake*	**60.5**	**67.2**	**56.1**

TABLE II: THE RESULT ON THE CITYSCAPES DATASET WHERE * MEANS THAT MULTI-COMPONENT DETECTION IS NOT APPLIED.

Name	AP
DANCE* [14]	36.7
Deep Snake [8]	37.4
E2EC [9]	39.0
TSnake	**39.5**

TABLE III: ABLATION STUDY OF THE TIME-EMBED SNAKE AND THE DILATED SNAKE. THE BOLD STYLE MEANS THE FIRST PLACE IN EACH EXPERIMENT.

Exp.	Method	AP_{vol}	AP_{50}	AP_{70}
Time - Embed Snake	w/o	58.9	65.9	54.8
	All layers	59.6	66.6	54.8
	backbone	**60.2**	**66.8**	**55.3**
Dilated Snake	w/o	59.9	66.2	55.1
	w/ Snake	59.9	66.5	55.0
	w/ Dilated	**60.2**	**66.8**	**55.3**
	Kernel=3	60.0	66.4	**55.4**
	Kernel=5	**60.2**	**66.8**	55.3

TABLE IV: ABLATION STUDY ON LOSS FUNCTIONS.

Experiment	Method	AP_{vol}	AP_{50}	AP_{70}
Loss Function	smooth L_1 loss	59.0	65.9	54.5
	DML	59.3	66.0	55.0
	Weighted DML	**60.2**	**66.8**	**55.3**
Weighted DML	After the 2nd	58.5	65.2	53.8
	After the 4th	**60.2**	**66.8**	**55.3**
	After the 6th	58.7	65.2	53.8

B. Implement Details

Fixed setting. We trained our models on Intel i9-9900K, one Nvidia RTX 4080, with Python 3.7, PyTorch 1.11, and cuda 11.3 version.

Backbone. For fair comparison to previous works, the detection backbone is CenterNet [20] with DLA34 [21] +DCNv2 [22].

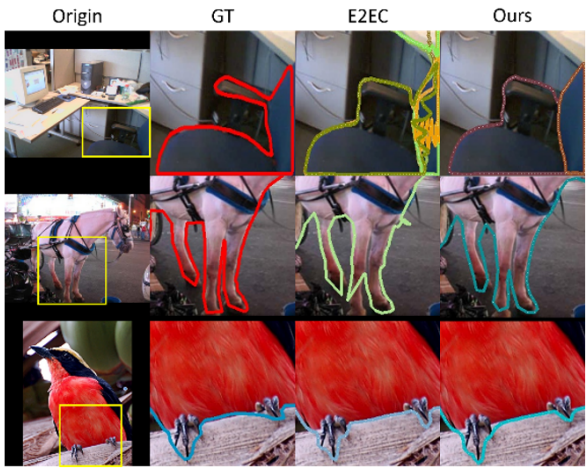

Fig. 5 From left to right are original images, ground truths, the results of E2EC [9] and our proposed method, respectively. Our method can provide contours that enclose the extreme points and not over-smoother than others.

Training setting. For SBD, TSnake was trained for 200 epochs with Adam optimizer. The learning rate starts from 1e-4 and decays based on StepLR at epochs 60, 100, 130, 155, 180. The batch size is set as 20. Mosaic augmentation [23] is adopted for the first 150 epochs, and the ratio of mosaic and original images is 3:2. The training and testing images are all 512x512.

For Cityscapes, multi-component detection [8] is applied for the separated objects. The optimizer and scheduler are the same as those in SBD but the learning rate decays at epochs 70, 110, 150, and 190.

C. Results

SBD. Our result of SBD is shown in Table I. TSnake outperformed the state-of-the-art (SOTA) contour-based method. It is worth mentioning that if we remove the dilated snake and refine the contours by time-embed snake only, our approach achieves 60.3 AP_{vol}, which yields a gap of 0.3 than SOTA. Meanwhile, our approach gets a balance between too smooth and too sensitive compared to and E2EC [9], represented in Fig. Some segmentation results are visualized in Fig. 4.

Cityscapes. Our result of Cityscapes is shown in Table II. By adopting multi-component detection, our TSnake gets 39.6 AP in validation set, which is the second place on the board compared to other contour-based methods. It yields 0.6 AP to the E2EC method [9]. Some results are shown in Fig. 5.

D. Ablation Study

In this section, we are going to verify the importance of each main component in our proposed method, including the time-embed snake, the dilated snake, and the weighted dynamic matching loss. Following [8], all the models will be trained with the SBD dataset [24] and do not conduct mosaic augmentation in ablations.

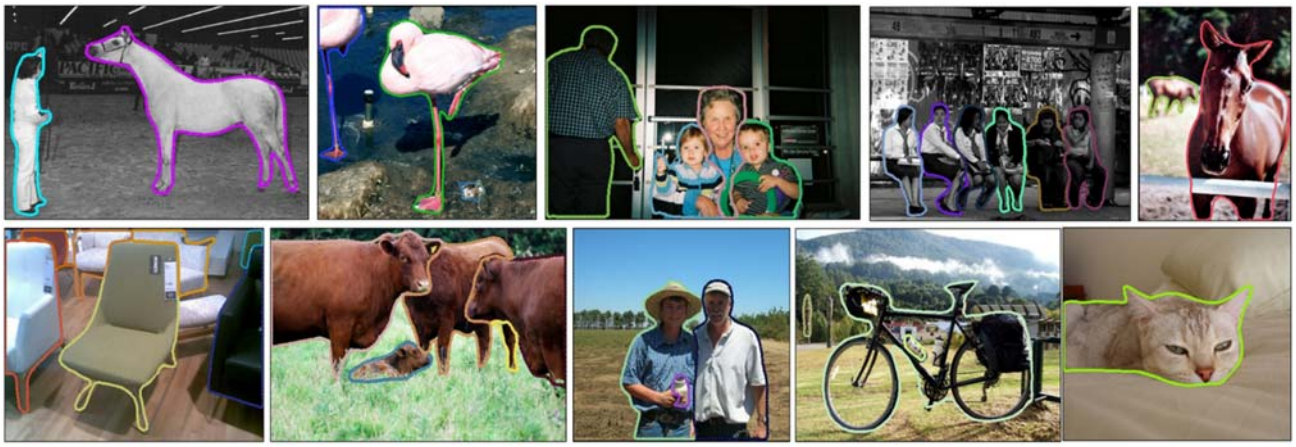

Fig. 6 The results on the SBD dataset.

Fig. 7 The results on the Cityscapes dataset.

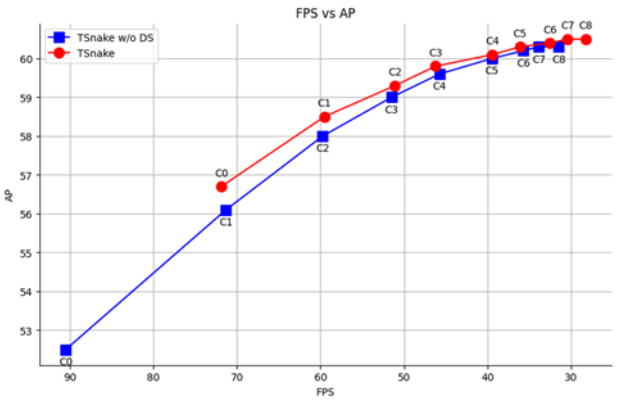

Fig. 8 Performance vs FPS in each iteration. The red and blue lines are the results of the TSnake with and without using the dilated snake, respectively. The performance converges at the 5th iteration, but still gets about 30 FPS. It shows that the proposed algorithm can be performed in real time.

Time-embed snake. We developed three settings to verify time embedding, shown in Table III. The first one is the origin snake module from Deep Snake [8], the second one will be added time embedding in every layer, and the third one is our proposed setting. Without time embedding, the origin snake got the worst performance with 1.3 AP_{vol} drop contrast to ours, which shows that time embedding really correctly controls the model in every iteration.

Dilated snake. Three cases are tested: (1) training without a dilated snake, (2) training by replacing a dilated snake with a snake module [8], and (3) training and testing with a dilated snake. The dilation of the kernel is set to 5, which is the same as our setting. The result is in Table III. Our dilated snake module got the best performance, yielding 0.3, 0.6 and 0.2 in AP_{vol}, AP_{50} and AP_{70} respectively compared to no addition one.

Loss function. Two experiments are conducted. The first is the importance of the weighted DML, and the second is which iteration is better to apply it. The results are shown in Table IV. In the first experiment, we replaced weighted DML by DML [9] and the smooth-L_1 loss. It shows that our proposed weighted DML improves by 0.9 AP_{vol} than the original DML, and yields 1.2 AP than the smooth-L_1 loss.

For the second experiment, we applied the weighted DML after the 2nd, 4th, and 6th iterations. As shown in Table IV, weighted DML started after the 4th iteration performs the best, yielding 1.0 AP_{vol} and 1.5 AP_{vol} to the 2nd and 6th iterations, respectively.

Iterative deformation. To show that the model is iteratively refined, we show the performance vs frame per second at each iteration, which is shown in Figs. 6 and 7.

Without the module, the performance significantly lagged behind the original in the first loops, however, by the 5th iteration, the gap narrowed to only 0.3 AP_{vol}, while the FPS increased by 3.5. An example of contour refinement at each iteration is shown in Fig. 8.

VI. Conclusion

In this work, we introduce TSnake, a time-embedded driven recurrent contour-based instance segmentation. TSnake applies time embedding to tell it which iteration is and deform the contours correspondingly. Next, the dilated snake shows that enriching receptive fields helps better reformation. Also, weighted DML reduces the sensitivity to noise, making it attractive to corners but not jagged. Our extensive experiments on the SBD and Cityscapes datasets demonstrate that TSnake achieves state-of-the-art performance while maintaining real-time efficiency.

References

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE Int. Conf. Computer Vision*, pp. 2961-2969, 2017.

[2] Z. Cai and N. Vasconcelos, "Cascade R-CNN: High quality object detection and instance segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, issue 5, pp. 1483-1498, 2019.

[3] X. Shen, J. Yang, C. Wei, B. Deng, J. Huang, X. S. Hua, X. Cheng, and K. Liang, "DCT-Mask: Discrete cosine transform mask representation for instance segmentation," in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp. 8720-8929, 2021.

[4] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *IEEE Int. Conf. Computer Vision*, pp. 9157-9166, 2019.

[5] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting objects by locations," in *European Conf. Computer Vision*, pp. 649-665, 2020.

[6] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen, and Ping Luo, "PolarMask: Single shot instance segmentation with polar representation," in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp. 12193-12202, 2020.

[7] W. Xu, H. Wang, F. Qi, and C. Lu, "Explicit shape encoding for real-time instance segmentation," in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp. 5168-5177, 2019.

[8] S. Peng, W. Jiang, H. Pi, X. Li, H. Bao, and X. Zhou, "Deep snake for real-time instance segmentation," in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp. 8533-8542, 2020.

[9] T. Zhang, S. Wei, and S. Ji, "E2EC: An end-to-end contour-based method for high-quality high-speed instance segmentation," in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp. 4443-4452, 2022.

[10] H. Feng, K. Zhou, W. Zhou; Y. Yin, J. Deng, Q. Sun, and H. Li, "Recurrent generic contour-based instance segmentation with progressive learning," *IEEE Trans. Circ. Syst. Video Tech.*, doi: 10.1109/TCSVT.2024.3383829. keywords: Apr. 2024.

[11] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, Jan. 1988.

[12] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid task cascade for instance segmentation," in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp. 4974-4983, 2019.

[13] X. Wang, F. Zhang, and Y. Huang, "Fourier contour embedding deep learning for arbitrary-shaped target detection," in *Advanced Fiber Laser Conference*, vol. 12595, pp. 303-307, 2023.

[14] Z. Liu, J. H. Liew, X. Chen, and J. Feng, "DANCE : A Deep attentive contour model for efficient instance segmentation," in *IEEE Winter Conf. Applications of Computer Vision*, pp. 345–354, 2021.

[15] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840-6851, 2020.

[16] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv*: 1511.07122, 2015.

[17] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE Int. Conf. Computer Vision*, pp. 2980-2988, 2017.

[18] 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 *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 3213–3223, 2016.

[19] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[20] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv*: 1904.07850, 2019.

[21] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp. 2403–2412, 2018.

[22] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets V2: More deformable, better results,". in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp. 9308-9316, 2019.

[23] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv*: 2004.10934, 2020.

[24] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Int. Conf. Computer Vision*, pp. 991–998, 2011.