Explicit Shape Encoding for Realtime Instance Segmentation

Wenqiang Xu Haiyang Wang Fubo Qi
Cewu Lu
Department of Computer Science and Engineering
Shanghai Jiao Tong University
{vinjohn, wanghaiyang, 727749815, lucewu}@sjtu.edu.cn

Abstract

In this paper, we propose a novel top-down instance segmentation framework based on explicit shape encoding, named ***ESE-Seg****. It largely reduces the computational consumption of the instance segmentation by explicitly decoding the multiple object shapes with tensor operations, thus performs the instance segmentation at almost the same speed as the object detection. ESE-Seg is based on a novel shape signature Inner-center Radius (IR), Chebyshev polynomial fitting and the strong modern object detectors. ESE-Seg with YOLOv3 outperforms the Mask R-CNN on Pascal VOC 2012 at* mAP^r*@0.5 and is also 7 times faster. It is adequate for many real applications. The code and trained model will be published with this paper.*

1. Introduction

Instance segmentation is a fundamental task in the computer vision, which is important for many real-world applications such as autonomous driving, robot manipulation. As the task seeks to predict both the object location and the shape, the methods for the instance segmentation are generally not as efficient as the object detection frameworks. Forwarding each object instance through an upsampling network to obtain the instance shape, as mainstream instance segmentation frameworks do [9, 19, 1, 16], is quite computation-consuming, especially when compared with the object detection which only needs to regress the bounding box, *i.e.* a 4D vector for each object. Thus, if the network can also regress the object shape to a short vector, and decode the vector to the shape (see Fig. 1) in a simple way just like the bounding box, it can make the instance segmentation reach *almost* equal computational efficiency to the object detection. To achieve this goal, we propose a novel instance segmentation framework based on **E**xplicit **S**hape **E**ncoding and modern object detectors, named **ESE-Seg**.

Shape encoding is originally developed for instance retrieval [32, 14, 30], which encodes the object to a shape vector. Recently, a number of works encode the shape implicitly [6, 24, 31], which is to project the shape content to a latent vector, typically through a black-box design such as deep CNN. Thus the decoding procedure under this approach should be also put through a network, which requires several forwarding for multiple instances, and causes large computation. In pursuit of fast decoding, we employ an explicit shape encoding that involves simple numeric transformations.

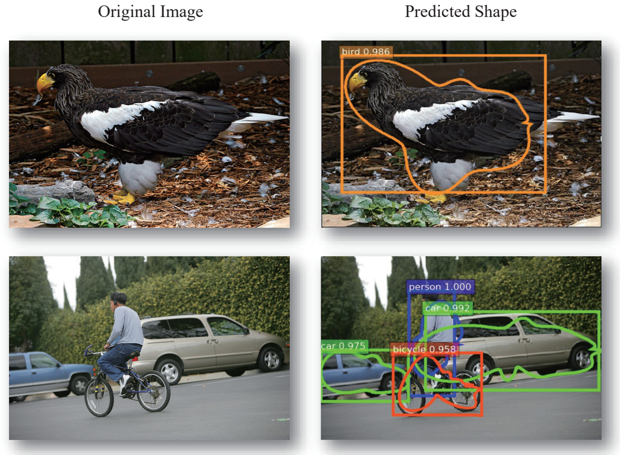


Figure 1. ESE-Seg learns to estimate the shapes of the detected objects, it can be simutaneously obtained along with the bounding boxes.

However, designing a satisfactory explicit shape encoding method is non-trivial. With respect to the CNN training, as it is known to regress with uncertainties, a preferred shape vector should be relatively **short** but contains sufficient information, **robust** to the noise, and **efficiently decodable** to reconstruct the shape. In this paper, we propose a contour-based shape signature to meet these requirements. A novel "Inner-center Radius" (IR) shape signature for instance shape representation is introduced. The IR first locates an inner-center inside the object segment, and based on this inner-center, it transforms the contour points to polar coordinates. That is, we can form a function of radius $\boldsymbol{f}(\theta)$ along the contour with respect to angle θ. To make the shape vector even shorter and more robust, we apply the Chebyshev polynomials to conduct the function approximation on $\boldsymbol{f}(\theta)$. As such, the IR signature is represented by a small number of coefficients with small error, and these coefficients are the shape vector to be predicted. Additionally, we also in-depth discuss about the comparison with other shape signature designs. Conventional object detector (*e.g*. YOLOv3 [26]) is used to regress the shape vector, along with 4D bounding box vector. To note that our shape decoding can be implemented by simple tensor operations (multiplication and addition) which are extremely fast.

The ESE-Seg itself is independent of all the bounding box-based object detection frameworks [27, 2, 5, 17, 20]. We demonstrate the generality on Faster R-CNN [27], RetinaNet [17], YOLO [25] and YOLO-tiny [26] and evaluate our ESE-Seg on standard public datasets, namely Pascal VOC [4] and COCO [18]. Our method achieves 69.3 mAP^r, 48.7 mAP respectively with IOU threshold 0.5. The score is better than Mask R-CNN [9] on Pascal VOC 2012, and is competitive to the performance on COCO. It is decent considering it is 7 times faster than Mask R-CNN with the same backbone ResNet-50 [10]. The speed can be even faster at 200fps on GTX 1080Ti when the base detector changes to YOLO-tiny, while the mAP remains 30.1% on the Pascal VOC. It is noteworthy, ESE-Seg speeds up the instance segmentation not depending on the model acceleration techniques [12, 33], but relying on a new mechanism that cut down shape prediction after object detection.

Contributions. We propose an explicit shape encoding based instance segmentation framework, ESE-Seg. It is a top-down approach but reconstructs the shapes for multiple instances in one pass, thus greatly reduces the computational consumption, and makes the instance segmentation reach the speed of the object detection with no model acceleration techniques involved.

2. Related Work

Explicit v.s Implicit Shape Representation A previous work with similar ideology has been done by Jetley *et al.* [13]. They took the implicit shape representation path by first training an autoencoder on object binary mask. The encoded shape vector is decoded to shape mask through the decoder component. In the implementation, they adopted the YOLO [25] to regress the bounding box and the shape vector for each detected object. The YOLO structure can thus be viewed as both detector and encoder. The encoded vector from YOLO is then decoded by the pre-trained denoising autoencoder. The major differences between our work and theirs:

- Explicit shape representation is typically based on the contour, while implicit shape representation is typically based on the mask.
- Explicit shape representation requires no additional decoder network training. Parallelizing the decoding process for all objects in the images, which is hard for network structured decoder, can be easily achieved by the explicit shape encoding. As a matter of fact, implicit decoding requires multiple pass for multiple objects, one for each, while explicit decoding can obtain all the shapes in one pass.
- The input for training autoencoder and training YOLO (viewed as encoder) is quite different (object scales, color pattern), which may cause trouble for the decoder, since the decoder is not further optimized with YOLO training. Such issue does not exist for explicit shape representation.
- Autoencoder is good for dimension reduction but not good for generalization.

In addition to our proposed IR shape signature, there exist various methods to represent the shape, to name a few, centroid radius, complex coordinates, cumulative angle [3, 28, 32] *etc.* While such methods sample the shape related feature along the contour, only a few of them can be decoded to reconstruct the shape.

Object detection Object detection is a richly studied field. Object detection frameworks with CNN can be roughly divided into two categories, one-stage and two-stage. Two-stage detection networks typically learn an RPN to sample region proposals and then refine the detection with roi pooling or its variations, the representative works are Faster R-CNN [27], R-FCN [2]. On the other hand, one-stage detectors divide the input image to size-fixed grid cells and parallelize the detection on each cell with fully convolutional operations, the representative networks are SSD [20], YOLO [25], RetinaNet [17]. Recently, point-based detections are proposed, CornerNet [15] directly detects the upper-left and bottom-right points, which is a one-stage detector. Grid R-CNN [21] regresses 9 points to construct the bounding box, which is a two-stage detector. Our method is compatible with all the bounding box-based detection networks. We experiment with Faster R-CNN, YOLOv1, YOLOv3, and RetinaNet to prove the generality. See Table 4. However, it is not compatible with the point-based detector, as the shape (bounding box) in this setting is not parametrized.

Instance Segmentation Instance segmentation requires not only to locate the object instance but also to delineate the shape. The mainstream methods can be roughly

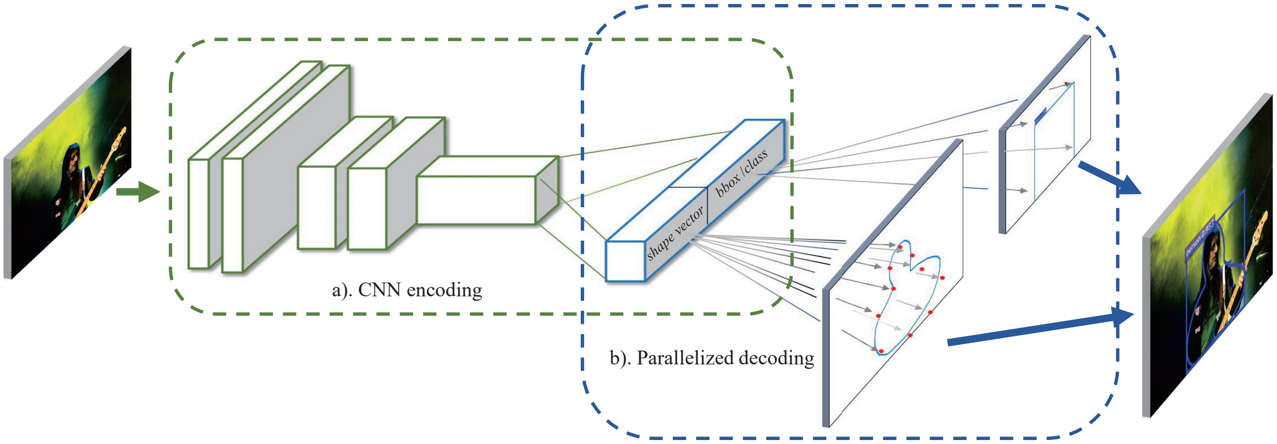


Figure 2. The pipeline of the shape detection, regression and reconstruction.

divided to top-down [9, 19, 1, 16] or bottom-up [23, 29] approaches. Ours belongs to the top-down line. The top-down approaches such as MNC [1], FCIS [16], Mask R-CNN [9] are generally slowed down when the object number in an image is large, as they predict the instance mask in sequence. On the contrary, our ESE-Seg alleviates the cumbersome computation by regressing the object shapes to short vectors and decoding them simultaneously. It is also the first top-down instance segmentation framework which is not affected by the instance number in the images with respect to the inference time.

3. Method

3.1. Overview

We propose an explicit shape encoding based detection to solve the instance segmentation. It predicts all the instance segments in one forwarding pass, which can reach equal efficiency as object detection solver. Given an object instance segment, we parametrize the contour with a novel shape signature "Inner-center Radius" (IR) (Sec. 3.2.1). The Chebyshev polynomials are used to approximate the shape signature vector with a small number of coefficients (Sec. 3.2.2). Those coefficients are served as the shape descriptor, and the network will learn to regress it. (Sec. 3.3). Finally, we describe how to decode the shape descriptor under the ordinary object detection framework by simple tensor operations. (Sec. 3.4). The overall pipeline is shown in Fig. 2.

The Advantage of Explicit Shape Encoding In object detection system (*e.g*. YOLOv3), the network regresses the bounding boxes (*i.e*. 4D vectors) and the bounding box is decoded by tensor operations, which is light to process and easy to parallelize. By contrast, conventional instance segmentation (*e.g*. Mask R-CNN) requires add-on network structure to compute the object shape. The decoding/upsampling forwarding involves a large amount of parameters, which is heavy to load in parallel for multiple instances. This is why instance segmentation is normally much slower than object detection. Therefore, if we also regress the object shape into short vectors directly, the instance shape decoding can be achieved by fast tensor operations (multiplication and addition) in a similar way. Thus the instance segmentation can reach the speed of object detection.

3.2. Shape Signature

3.2.1 Inner-center Radius Shape Signature

In this section, we will describe the design of the "inner-center radius" shape signature and compare it to previously proposed shape signatures.

The construction of the "inner-center radius" contains two steps: First, locate an Inner center point inside the object segment as the origin point to build the polar coordinate system. Second, sampling the contour points according to the angle θ. This signature is translation invariant and scale invariant after normalized.

Inner center The inner-center point is defined by the most far-way point from the contour, which can be obtained through distance transform [22]. To note, some commonly used center such as the center of mass, the center of the bounding box cannot guarantee to be inside the object. See Fig. 3.

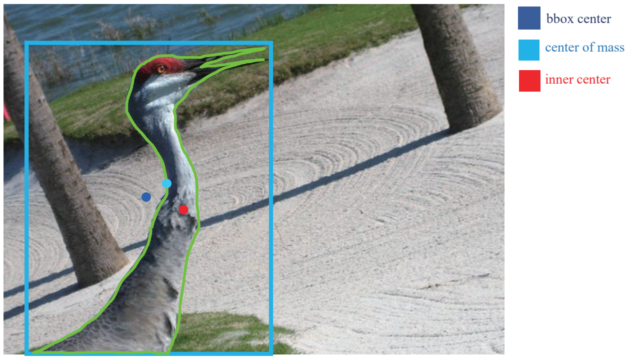


Figure 3. The center points of an object. As we can see, bbox center and the center of mass cannot gaurentee to be inside an object.

In a few cases, an object is separated into disconnected regions, resulting in multiple inner centers. To deal with such situations, we dilate the broken areas to a single one and then find the contour of the dilated shape. Of course the contour is very rough, however, it can help to reorder the contour points of the outline points. The whole process is depicted in Fig. 4. Thus inner center is computed from the completed contour.

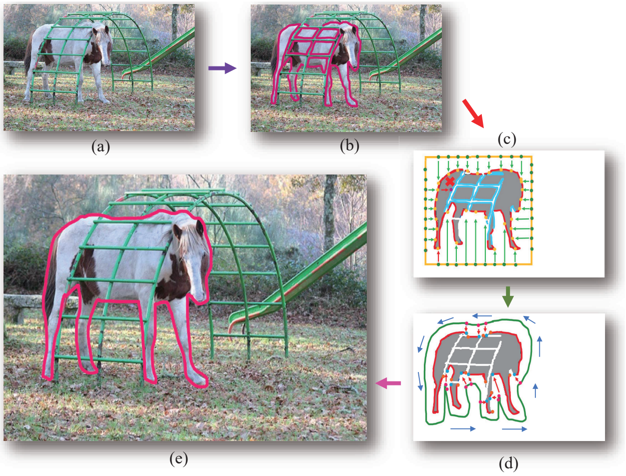


Figure 4. The process to complete the separated areas. An occluded object (a) has many separated areas (b), we split the contour points of each area into outline and inner points with the help of the bounding box (c), then we dilate the broken area into one, and reorder the outline points according to the dilated shape contour (d), finally, we complete the instance (e).

Dense Contour Sampling We sample the contour points according to the angles at the interval of τ around inner-center point, thus a contour will result in $N = [2\pi/\tau]$ points. In practice, $\tau = \pi/180$ and thus $N = 360$ points are sampled from an object contour. If the ray casting from the inner-center intersects more than once to the contour. We collect the point with the largest radius only. The function $\boldsymbol{f}(\theta)$ is denoted as radius at different angles θ. To note, we are aware that the contour sampling in this way will not be perfect, however, after extensive experiments in Pascal VOC, and COCO, we find it suitable for natural objects (see Table 2). A further discussion is in the next Sec. 3.2.3.

3.2.2 Fitting the Signature to Coefficients

The IR makes shape representation into a vector. But, it is still too long for the network to train. Besides, the shape signature is very sensitive to the noise (see Fig. 7). Thus, we take a further step to shorten shape vector and resist noise through Chebyshev polynomial fitting.

Chebyshev polynomials The Chebyshev polynomial is defined in recurrence:

$$T_0(x) = 1, \tag{1}$$

$$T_1(x) = x, \tag{2}$$

$$T_{n+1}(x) = 2xT_n(x) - T_(n-1)(x), \tag{3}$$

which is also known as **The Chebyshev polynomials of the first kind**. It can effectively minimize the problem of *Runge's phenomenon* and provides a near optimal approximation under the maximum norm[1].

Given the IR shape signature, the Chebyshev approximation is to find the coefficients in

$$\boldsymbol{f}(\theta) \sim \sum_{i=0}^{\infty} c_i T_i(\theta)$$

Truncating the function with n terms, we have the approximation function $\tilde{f}(\theta) = \sum_{i=0}^{n} c_i T_i(\theta)$. $\boldsymbol{c} = (c_0, \dots, c_n)$ are the shape signature vector to represent the object.

3.2.3 Discussion

Comparison with Other Shape Signatures The angle-based sampling for shape signature such as proposed IR is rarely adopted before, because it cannot perfectly fit shape segment. Actually we compare and in-depth analyze other shape signatures and finally choose this solution. For example, a quite straight-forward design is to sample along the contour. The contour is represented by a set of contour polygon vertex coordinates. This method can nearly perfectly fit object segment especially non-convex shape. However, we find the performance of this design drops about 10 mAP and more results are reported in Table 2. The possible reason is that our angle-based sampling produces 1D sample

[1] https://en.wikipedia.org/wiki/Chebyshev_polynomials

sequence, yet, contour vertices sequence is 2D sample sequence which is more sensitive to noise. We report the reconstruction error of these two shape signatures on Pascal VOC 2012 training in Fig. 5 (denoted as “IR” and “XY” respectively). Admittedly, the XY has less reconstruction error when sampling the same points on the contour, but when compared with the same dimension of the vector, IR is more accurate. For example, the dimension of the vector of IR at $N = 20$ is the same as XY at $N = 10$, the IR has a significantly less reconstruction error. Though when the N gets larger, the difference gets smaller, but a large N will make training unstable as presented in Table 2.

Other classic shape signatures such as centroid radius, cumulative angle cannot reconstruct the shape.

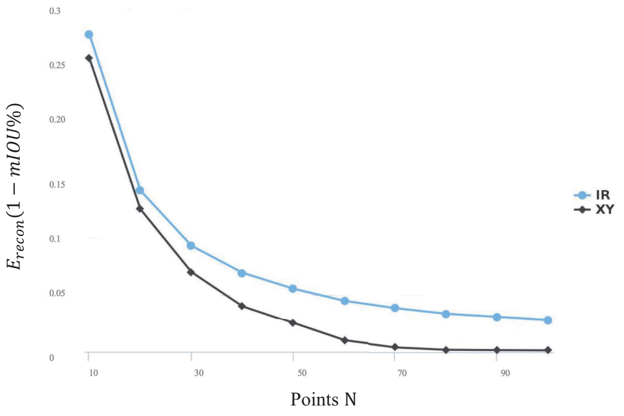


Figure 5. The reconstruction error E_{recon} of IR and XY with different sampling number points.

Comparison with Other Fitting Methods Other commonly used function approximation methods, namely polynomial regression and Fourier series fitting are also considered.

For polynomial regression, the goal is to fit shape vector $\boldsymbol{k} = (v_0, \dots, v_n)$, which is the coefficients of n degree polynomials, $\tilde{\boldsymbol{f}}(\theta) = \sum_{i=0}^{n} v_i x^i$. For Fourier series fitting, the shape vector is $\boldsymbol{k} = (\omega, a_0, a_1, \dots, a_n, b_1, \dots, b_n)$, the truncated n degree Fourier series is $\tilde{\boldsymbol{f}}(\theta) = a_0/2 + \sum_{i=1}^{n}[a_i \cos(i\omega\theta) + b_i \sin(i\omega\theta)]$. As the dimension of $\boldsymbol{k}$ can be determined in advance, denoted as l. Thus we compare the methods from three aspects, *i.e.* the reconstruction error E_{recon}, sensitivity to the noises, and the numeric distribution of the coefficients.

The reconstruction errors E_{recon} is calculated by $1 - mIOU$ under the same dimension l and point number N in Fig. 6. Then we set $l = 8$ as an example to conduct the sensitivity analysis as shown in Fig. 7. For each coefficient, it is interrupted by the noise $\varepsilon \sim N(0, \alpha\bar{k})$, $\bar{k}$ is the mean of the corresponding coefficient. As we can see, the ω of Fourier series is extremely sensitive, which may cause the Fourier fitting not suitable for the CNN training, as the CNN is known to regression with uncertainties. If we fix $\omega = 1$, it becomes less sensitive, but has considerably larger reconstruction error. Besides, considering the difficulty for network to learn, we also investigate the statistic on the distribution of the fitted coefficients. See Fig. 9, Fig. 10 and Fig. 11. Chebyshev polynomials is better for shape signature fitting as it has less reconstruction error, less sensitivity to noise, better numeric distribution of coefficients.

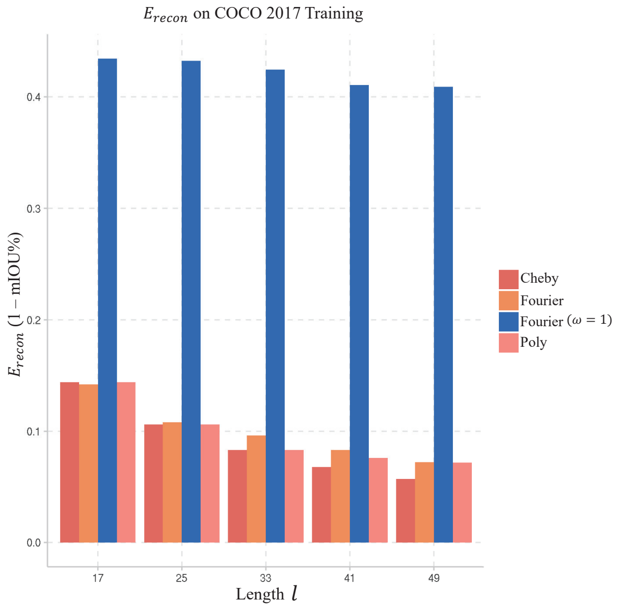


Figure 6. Comparison of E_{recon} on COCO 2017 training.

3.3. Regression Under Object Detection Framework

Our network will learn to predict the inner center $\hat{\boldsymbol{p}}$, the shape vector $\hat{\boldsymbol{c}}$, along with the object bounding box. The loss function for bounding boxes regression, classification stay the same to the original object detection frameworks. For YOLOv3, the loss function for bounding box $\mathcal{L}_{bbox}$ and classification $\mathcal{L}_{cls}$ can be referred to [26]. As for the loss function for the shape learning:

$$\mathcal{L}_{shape} = \mathbb{1}^{obj} ||(\hat{\boldsymbol{p}} - \boldsymbol{p}) + (\hat{\boldsymbol{c}} - \boldsymbol{c})||_2^2,$$

where $\mathbb{1}^{obj}$ indicates the grid cells with objects for the one-stage detectors, and the proposals for the two-stage detectors. Thus the overall objective function is:

$$\mathcal{L} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{bbox}\mathcal{L}_{bbox} + \lambda_{shape}\mathcal{L}_{shape}$$

3.4. Decoding Shape Vector to Shape

Given the shape vector dimension l, the predicted shape vector $\hat{\boldsymbol{c}} = (\hat{c}_0, \dots, \hat{c}_{l-1})^\top$, the fitted Chebyshev polyno-

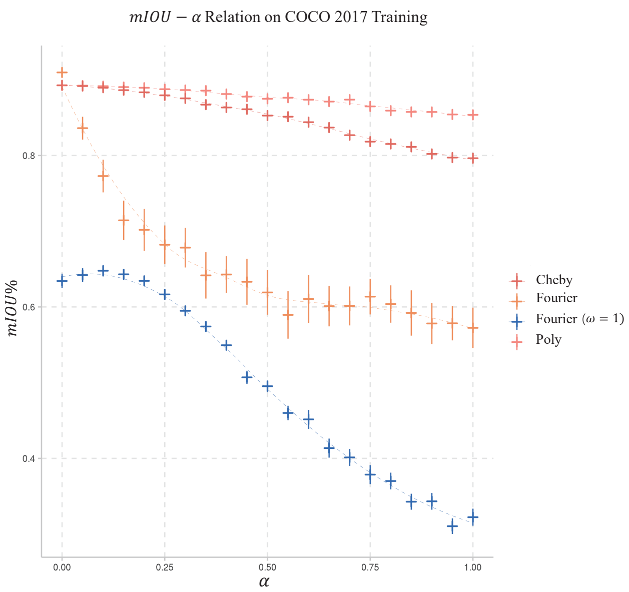


Figure 7. Comparison of the sensitivity on COCO 2017 training.

mial is $\hat{f}(\theta) = \sum_{i=0}^{l-1} \hat{c}_i T_i(\theta)$. And the polar coordinate transform factor $\boldsymbol{u}(\theta) = (\cos\theta, \sin\theta)$.Thus the shape can be recovered by traversing the $\theta \in [0, 2\pi)$

$$\hat{\boldsymbol{p}}_i = \hat{\boldsymbol{p}}_c + \hat{\boldsymbol{f}}(\theta) \odot \boldsymbol{u}(\theta).$$

$\odot$ is the Hadamard product. This calculation can be written in tensor operation form. Given the batch size bs, the corresponding tensor version are $\boldsymbol{\Theta} \in \mathbb{R}^{bs\times 1\times N}$ for angles sampled, $\hat{\boldsymbol{C}} \in \mathbb{R}^{bs\times 1\times l}$ for the predicted shape vector, $\hat{\boldsymbol{P}}_{\boldsymbol{c}} \in \mathbb{R}^{bs\times 2\times N}$ for the predicted inner centers and $\hat{\boldsymbol{P}} \in \mathbb{R}^{bs\times 2\times N}$ represents the decoded contour points. As expressed:

$$\hat{\boldsymbol{P}} = \hat{\boldsymbol{P}}_c + \hat{\boldsymbol{C}}T(\boldsymbol{\Theta}) \odot \boldsymbol{u}^\top(\boldsymbol{\Theta}).$$

In the GPU setting, the computation cost of such tensor operation is very minor. Due to this extremely fast shape decoding, our instance segmentation can achieve the same speed with object detection.

4. Experiment

We conduct extensive experiments to justify the descriptor choice and the efficacy of proposed methods. If not specified, the base detector is YOLOv3 implemented by GluonCV [11], the input image is 416 × 416. $\lambda_{cls} = \lambda_{bbox} = \lambda_{shape} = 1$. Other hyper-parameters stays the same as the YOLOv3 implementation. We trained 300 epochs and report the performance with the best evaluation results. For the model name with a bracket and a number in it, the number is the dimension of the shape vector.

4.1. Explicit v.s. Implicit

We first compare the explicit shape encoding with the implicit shape encoding. As the previous work [13] provides a baseline for implicit shape representation with YOLO [25] as the base detector, to be fairly compared, we also trained the ESE-Seg with YOLO base detector, the dimension of the shape vector is also the same. We denote the model as "YOLO-Cheby (50)" and "YOLO-Cheby (20)". The experiments are on Pascal SBD 2012 val [7].

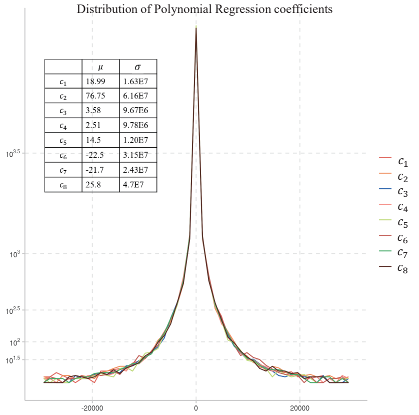
To note, the mainstream instance segmentation based on mask, namely SDS [8], MNC [1], FCIS [16], Mask RCNN [9], can also be viewed as implicit shape encoding. We compare them with "YOLOv3-Cheby (20)" on Pascal VOC 2012 without SBD and COCO with their reported scores, which outperforms the Mask R-CNN (with ResNet50) at mAP^r@0.5 on Pascal VOC and close to it on COCO. To note, the input image size is 800 on the shorter side for Mask R-CNN with ResNet50-FPN, which is almost 4 times to our 416 × 416. All results are reported in Table 1.

SBD (5732 val images)			
model	mAP^r@.5	mAP^r@.7	Time (ms)
BinaryMask[13]	32.3	12.0	26.3
Radial[13]	30.0	6.5	27.1
Embedding (50) [13]	32.6	14.8	30.5
Embedding (20) [13]	34.6	**15.0**	28.0
YOLO-Cheby (50)	39.1	10.5	24.2
YOLO-Cheby (20)	**40.7**	12.1	**24.0**
Pascal VOC 2012 val			
model	mAP^r@.5	mAP^r@.7	Time (ms)
SDS [8]	49.7	-	48k
MNC	65.0	46.7	330
FCIS	65.7	**52.1**	-
Mask R-CNN	68.5	40.2	200
YOLO3-Cheby (20)	62.6	32.4	**29.0**
+ COCO pretrained	**69.3**	36.7	**29.0**
COCO 2017 val			
model	mAP@50	mAP@75	Time (ms)
FCIS	49.5	-	-
Mask R-CNN	51.2	31.5	200
YOLO3-Cheby (20)	48.7	22.4	**29.0**

Table 1. Comparison of ESE-Seg to the previous methods on Pascal SBD 2012 val, Pascal VOC 2012 without SBD val, and COCO 2017 val.

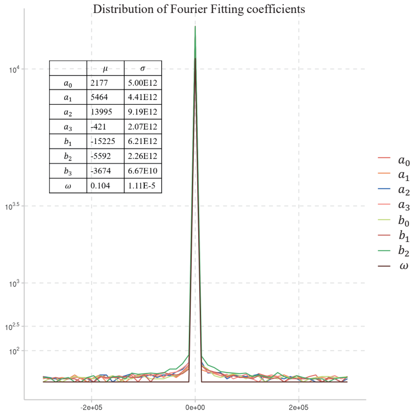
4.2. On explicit descriptors

In this section, we will compare the object shape signatures and the function approximation methods quantitatively.



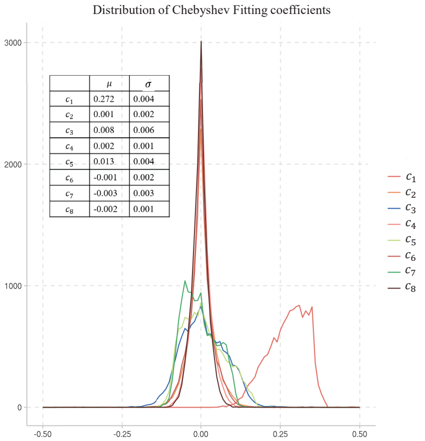
	μ	σ
c_1	18.99	1.63E7
c_2	76.75	6.16E7
c_3	3.58	9.67E6
c_4	2.51	9.78E6
c_5	14.5	1.20E7
c_6	-22.5	3.15E7
c_7	-21.7	2.43E7
c_8	25.8	4.7E7

Figure 9. The overall mean of the coefficients is , and the variance is for Polynomial regression.



	μ	σ
a_0	2177	5.00E12
a_1	5464	4.41E12
a_2	13995	9.19E12
a_3	-421	2.07E12
b_1	-15225	6.21E12
b_2	-5592	2.26E12
b_3	-3674	6.67E10
ω	0.104	1.11E-5

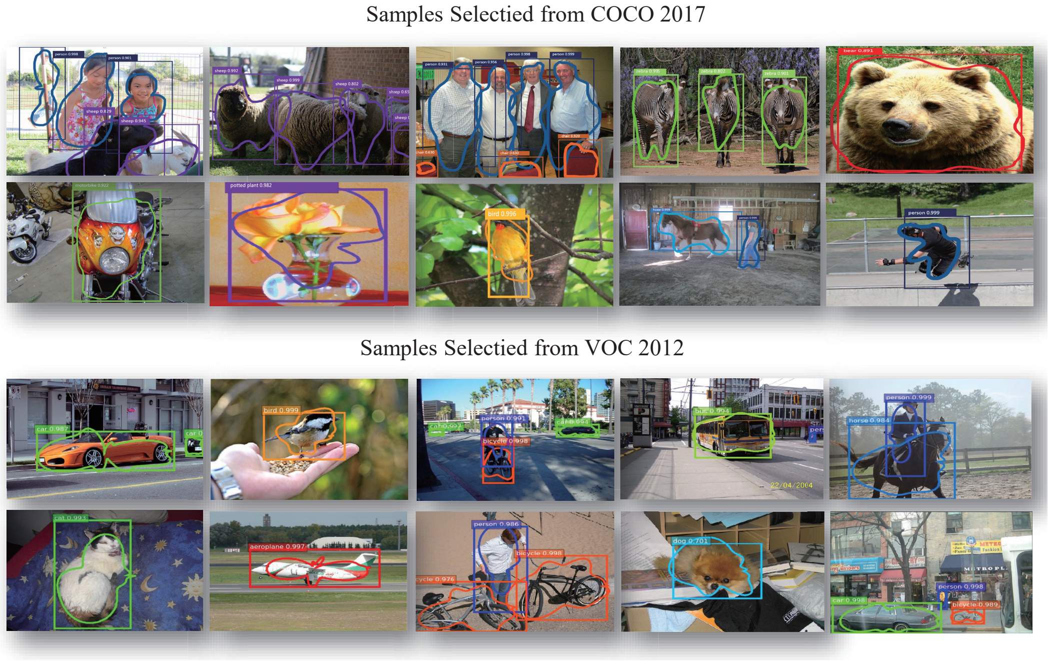
Figure 10. Coefficients distribution of Fourier series fitting on COCO training 2017.



	μ	σ
c_1	0.272	0.004
c_2	0.001	0.002
c_3	0.008	0.006
c_4	0.002	0.001
c_5	0.013	0.004
c_6	-0.001	0.002
c_7	-0.003	0.003
c_8	-0.002	0.001

Figure 11. Coefficients distribution of Chebyshev polynomial fitting on COCO training 2017.

Figure 11. Coefficients distribution of Polynomial regression, Fourier Series fitting, Chebyshev Polynomial fitting.


Samples Selectied from COCO 2017

Samples Selectied from VOC 2012

Figure 12. Qualitative results generated by our methods.

On Different Shape Signatures For object shape signatures, we compare our proposed IR with a straightforward 2D vertices representation on Pascal VOC 2012. (See Table 2) We adopt the squared boxes, *i.e*. the bounding box, as the baseline. To note, the squared boxes baseline is not the object detection scores, as the baseline computes the IoU between the bounding box and the instance mask.

For each shape signature, we compare regressing directly and regressing after Chebyshev polynomial fitting. For direct regression, we control the length of the shape signature by adjusting the τ for each shape. We select 20 and 40 points to regress. We denote model trained on 2D vertices "XY", the shape vector has a dimension of 40 and 80 respectively. As for the Chebyshev fitting on these signatures, we fit the x coordinates and y coordinates respectively. Denoted as "XY-Cheby (10+10)" means each fitted function

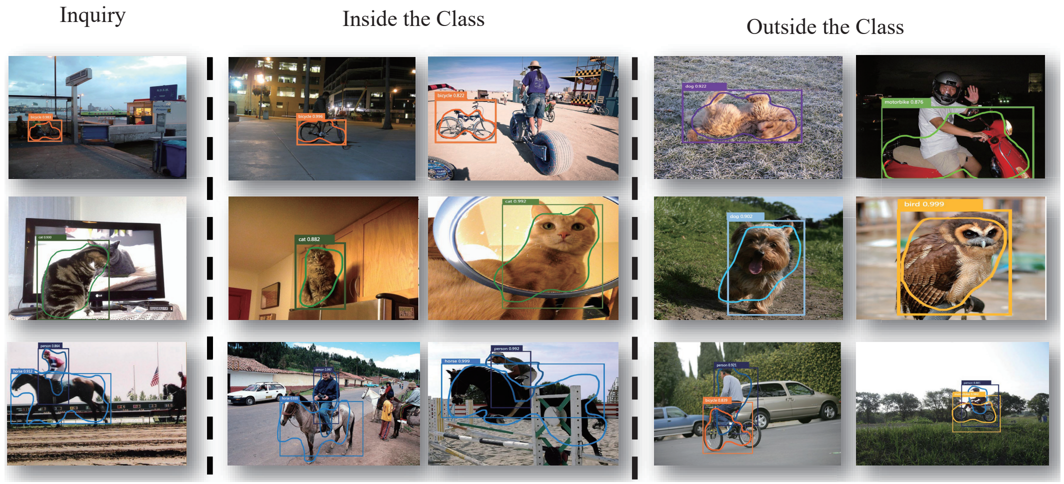


Figure 13. The learned shape vector used for instance retrieval. It just a showcase.

has 10 coefficients.

model	mAP^r@.5	mAP^r@.7
Squared Boxes	42.3	8.6
XY (20)	46.1	10.7
XY (40)	43.5	11.2
XY-Cheby (10+10)	48.3	16.4
XY-Cheby (20+20)	53.1	20.9
IR (20)	48.8	13.5
IR (40)	52.6	19.3
IR (60)	51.7	16.4
IR-Cheby (20)	62.6	32.4

Table 2. We compare different choice of the shape signatures on Pascal VOC 2012.

On Different Function Approximation Techniques We have already compared the function approximation techniques through off-line analysis. However, it is still interesting to known performance of the neural network on the coefficients obtained by these methods.

All the function approximations are carried out on IR $f(\theta)$. The polynomial regression is denoted as "Poly", while "Fourier" for Fourier series fitting and "Cheby" for Chebyshev polynomial fitting. All models have tested on Pascal VOC 2012 val. See Table 3.

4.3. On base object detector

To show the generality of the object shape detection, we also conduct the shape learning on Faster R-CNN ("Faster-Cheby (20)"), RetinaNet ("Retina-Cheby (20)") and YOLO-tiny ("YOLO-tiny-Cheby (20)"). Not only the performance is stable for all these bounding box-based detectors, but the speed boost due to the detector can be enjoyed. As shown in Table 4.

model	mAP^r@.5	mAP^r@.7
Poly (20)	26.3	5.4
Fourier (20)	37.5	9.1
Fourier (40)	36.1	8.5
Cheby (20)	62.6	32.4
Cheby (40)	60.7	31.5

Table 3. Comparison of the performance of different shape signatures on Pascal VOC 2012 val.

model	mAP^r@.5	mAP^r@.7	Tims (ms)
YOLO3-Cheby (20)	62.6	32.4	29.0
Faster-Cheby (20)	63.4	32.8	420
Retina-Cheby (20)	65.9	36.5	73
YOLO-tiny-Cheby (20)	30.1	5.4	5.2

Table 4. Comparison of different base object detectors with IR shape signature and Chebyshev fitting on Pascal VOC 2012 val.

4.4. Qualitative Results

Qualitative results are shown in Fig. 12. Obviously, the predicted shape vectors indeed capture the characteristics of the contours, not produce the random noise.

4.5. Instance Retrieval

The predicted shape vector can be directly used for instance retrieval. We give some examples on the Pascal VOC 2012 as shown in Fig. 13. It supports to search instance of similar shape with different scales, different location, and different combinations. It is an interesting bonus result, and we will not evaluate on this task, but just present it as a demonstration.

5. Limitations and Future Works

Our proposed ESE-Seg can achieve the instance segmentation with minor time-consumption, with a decent performance at IoU threshold 0.5. However, due to the inaccuracy E_{recon} of the shape vector, and the noise comes with the CNN regression, performance at larger IoU threshold like 0.7 drop a large margin. To train the CNN regression with stable output is the future work.

References

[1] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

[2] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[3] E. Davies. Machine vision: Theory, algorithms and practicalities. 1990, 1997.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[5] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[6] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *European conference on computer vision*, pages 241–257. Springer, 2016.

[7] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011.

[8] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. *arXiv preprint arXiv:1812.01187*, 2018.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[13] S. Jetley, M. Sapienza, S. Golodetz, and P. H. Torr. Straight to shapes: real-time detection of encoded shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6550–6559, 2017.

[14] H.-K. Kim and J.-D. Kim. Region-based shape descriptor invariant to rotation, scale and translation. *Signal Processing: Image Communication*, 16(1):87 – 93, 2000.

[15] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[16] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367, 2017.

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

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

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[21] X. Lu, B. Li, Y. Yue, Q. Li, and J. Yan. Grid r-cnn. *CoRR*, abs/1811.12030, 2018.

[22] C. R. Maurer, R. Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[23] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 2274–2284, USA, 2017. Curran Associates Inc.

[24] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki. Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications*, 4(3):251–258, 2016.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

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

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

[28] P. J. Van Otterloo. *A contour-oriented approach to shape analysis*. Prentice Hall International (UK) Ltd., 1991.

[29] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244, June 2009.

[30] I. T. Young, J. E. Walker, and J. E. Bowie. An analysis technique for biological shape. i. *Information and Control*, 25(4):357 – 370, 1974.

[31] J. Yue-Hei Ng, F. Yang, and L. S. Davis. Exploiting local features from deep networks for image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 53–61, 2015.

[32] D. Zhang, G. Lu, et al. A comparative study of fourier descriptors for shape representation and retrieval. In *Proc. 5th Asian Conference on Computer Vision*, page 35. Citeseer, 2002.

[33] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.