Applying Eigencontours to PolarMask-Based Instance Segmentation

Wonhui Park
Korea University
whpark@mcl.korea.ac.kr

Dongkwon Jin
Korea University
dongkwonjin@mcl.korea.ac.kr

Chang-Su Kim
Korea University
changsukim@korea.ac.kr

Abstract

Eigencontours are the first data-driven contour descriptors based on singular value decomposition. Based on the implementation of ESE-Seg, eigencontours were applied to the instance segmentation task successfully. In this report, we incorporate eigencontours into the PolarMask network for instance segmentation. Experimental results demonstrate that the proposed algorithm yields better results than PolarMask on two instance segmentation datasets of COCO2017 and SBD. Also, we analyze the characteristics of eigencontours qualitatively. Our codes are available at https://github.com/dnjs3594/Eigencontours.

1. Introduction

Instance segmentation is a task to delineate the shape of each distinct object in an image. Most instance segmentation techniques are based on the framework of object detection, such as Mask R-CNN [6] and its variants [2, 4, 10]. They detect the bounding box of each instance and then classify whether each pixel within the box belongs to the instance or not. However, despite yielding promising results, they often demand high computational costs due to the pixelwise classification.

To alleviate the computational complexity, contour-based techniques [11–13] also have been developed. These techniques reformulate the pixelwise classification task as the boundary regression task of an object. To this end, they represent object boundaries with contour descriptors. For example, in ESE-Seg [13], the boundary of an object is represented by polynomial fitting coefficients. Also, PolarMask [12] describes an object boundary in polar coordinates using centroidal profiles. Then, these techniques localize objects by predicting the contour description vectors only, which is computationally more efficient than the pixelwise classification. However, they may fail to reconstruct object boundaries accurately.

Recently, in [11], we proposed eigencontours, which are data-driven contour descriptors obtained by the low-rank approximation of all object boundaries in a training set through singular value decomposition (SVD). Experimental results showed that an object boundary can be described faithfully by a linear combination of a small number of eigencontours, and that eigencontours are more effective and more efficient than the conventional contour descriptors [12, 13]. Also, it was shown that, based on the framework of ESE-Seg, eigencontours improve the instance segmentation performance on the SBD dataset [5].

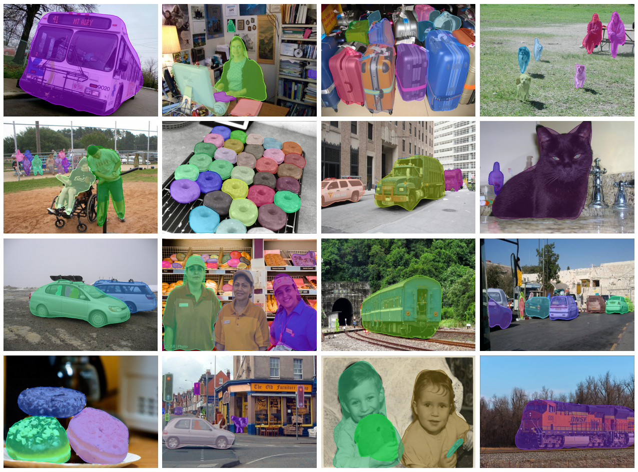
Figure 1. Instance segmentation results of the proposed algorithm on the COCO2017 validation dataset [9].

In this report, based on the framework of PolarMask [12], we apply eigencontours to the instance segmentation task. First, we describe how to construct eigencontours in detail. Then, we design a network by modifying the PolarMask network. It is shown experimentally that the proposed network outperforms PolarMask on the COCO2017 [9] and SBD [5] datasets. We also discuss the strengths and weaknesses of eigencontour descriptors qualitatively.

2. Eigencontour Representation

Let us describe how to represent object boundaries in the eigencontour space [11]. First, a star-convex contour is generated from each object boundary in a training set. Second, a star-convex contour matrix is constructed to include all the star-convex contours in the training set as its column vectors. Third, the star-convex contour matrix is decomposed into eigencontours via SVD. Last, an object boundary is represented by a linear combination of the first M eigencontours.

Star-convex contour generation: In [11], object boundaries are first simplified to star-convex contours to strike a balance between the accuracy and the simplicity of contour representation. To this end, centroidal profiles [3] are employed. Given an object shape, a center point $\mathbf{o} = (x, y)$ is defined, which can be determined as the inner-center [13] or the centroid [12]. Then, the boundary is described using polar coordinates (r_i, θ_i), $i = 1, 2, \ldots, N$. The angular coordinates θ_i are sampled uniformly, so only the radial coordinates are used to represent the contour

$$\mathbf{r} = [r_1, r_2, \ldots, r_N]^\top \tag{1}$$

where r_i is the distance of the farthest object point from the center along the θ_i-axis. Thus, a star-convex contour is expressed by $\mathbf{r}$.

Eigencontour construction: Let $\mathbf{A} = [\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_L]$ be a star-convex contour matrix, containing L star-convex contours of objects in a training set. Then, by performing SVD, the contour matrix $\mathbf{A}$ is decomposed into

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \tag{2}$$

where $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_N]$ and $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_L]$ are orthogonal matrices and $\mathbf{\Sigma}$ is a diagonal matrix, composed of singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. Here, r is the rank of $\mathbf{A}$. Then, the best rank-M approximation of $\mathbf{A}$ [1] is given by

$$\mathbf{A}_M = [\tilde{\mathbf{r}}_1, \cdots, \tilde{\mathbf{r}}_L] = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top + \cdots + \sigma_M \mathbf{u}_M \mathbf{v}_M^\top. \tag{3}$$

The first M left singular vectors $\mathbf{u}_1, \cdots, \mathbf{u}_M$ are referred to as *eigencontours*. Also, the space spanned by $\{\mathbf{u}_1, \cdots, \mathbf{u}_M\}$ is called the *eigencontour space*.

Contour representation: We can approximate a contour $\mathbf{r}$ by a linear combination of the first M eigencontours,

$$\tilde{\mathbf{r}} = \mathbf{U}_M \mathbf{c} = [\mathbf{u}_1, \cdots, \mathbf{u}_M]\mathbf{c} \tag{4}$$

where the coefficient vector $\mathbf{c}$ is given by

$$\mathbf{c} = \mathbf{U}_M^\top \mathbf{r}. \tag{5}$$

This means that an N-dimensional contour $\mathbf{r}$ is optimally approximated by an M-dimensional vector $\mathbf{c}$ in the eigencontour space, where $M < N$. Also, the approximate $\tilde{\mathbf{r}}$ can be reconstructed from $\mathbf{c}$ via (4). **Algorithm 1** summarizes how to construct the eigencontour space.

Algorithm 1 Eigencontour construction

```
Input: Set of training objects $\{\mathbf{x}_1, \cdots, \mathbf{x}_L\}$, $N$ = # of star-convex contour points, $M$ = # of eigencontours.
Generate $N$-dimensional star-convex contour vectors $\{\mathbf{r}_1, \cdots, \mathbf{r}_L\}$ and center points $\{\mathbf{o}_1, \cdots, \mathbf{o}_L\}$ for all training objects
Construct the star-convex contour matrix $\mathbf{A}$ and apply SVD in (2)
Perform the best rank-$M$ approximation and obtain the first $M$ eigencontours via (3)
Approximate each contour $\mathbf{r}_i$ to $\mathbf{c}_i$ via (4)
Output: Set of coefficients $\{\mathbf{c}_1, \cdots, \mathbf{c}_L\}$ and center points $\{\mathbf{o}_1, \cdots, \mathbf{o}_L\}$
```

3. Instance Segmentation Using Eigencontours

We apply eigencontours into instance segmentation based on the implementation of PolarMask [12].

Architecture: For instance segmentation, we design a network based on PolarMask [12]. Figure 2 shows the structure of the proposed network, which is composed of a single encoder and three decoders. Given an image, the encoder extracts a convolutional feature map of size $H \times W \times C$. Then, from the feature map, the three decoders generate output maps, respectively: $P \in \mathbb{R}^{H \times W \times K}$, $O \in \mathbb{R}^{H \times W \times 1}$, and $R \in \mathbb{R}^{H \times W \times M}$. Each element in P is a K-dimensional vector, indicating whether the corresponding pixel belongs to an object in one of the K predefined categories. Each score in O represents the probability that the corresponding pixel is the center point of an object. Each element in R is a M-dimensional vector, which, in the eigencontour space, represents the contour of the object centered on the corresponding pixel. As in [12], the encoder is based on the feature pyramid network with the ResNet50 backbone [7]. Also, each decoder is implemented by a series of 2D convolution layers.

Coefficient regression: In PolarMask, an object boundary in an image is represented by a star-convex contour. To detect the object boundary, it should regress N variables in $\mathbf{r}$ in (1). On the other hand, in this work, an object contour is represented by a linear combination of the M eigencontours. Thus, the proposed network needs to regress only the M coefficients of $\mathbf{c}$ in (5) in the eigencontour space.

Postprocessing: From the output maps, we extract the segmentation masks of objects also based on the postprocessing scheme of PolarMask. For each pixel i, we first compute a confidence score by multiplying the score O_i with the maximum probability in P_i. We then filter out the pixels with confidence scores lower than 0.05. Then, we perform non-maximum suppression. Specifically, we iteratively select the pixel with the maximum confidence score

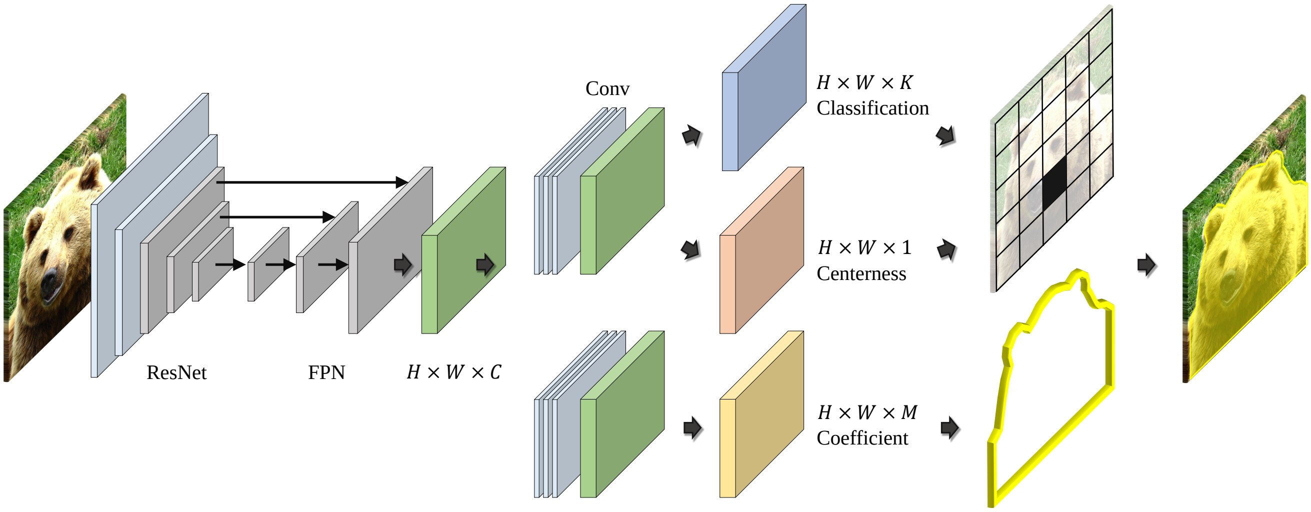

Figure 2. The architecture of the proposed instance segmentation network based on PolarMask.

and remove the pixels whose decoded masks overlap with the mask of the selected pixel. Finally, we declare the segmentation masks of the selected pixels as the final results. Notice that, to get a segmentation mask, we reconstruct the contour vector by linearly combining the eigencontours using the coefficients in R_i. Then, we convert the contour vector to contour points using the uniformly sampled angular coordinates.

4. Experiments

4.1. Datasets

We conduct experiments on the COCO2017 [9] and SBD [5] datasets. COCO2017 consists of 118K training and 5K validation images, and SBD contains 5,623 training and 5,732 validation images. For COCO2017 and SBD, instance segmentation masks are provided for objects in $K = 80$ and 20 categories, respectively.

4.2. Training details

We use the stochastic gradient descent (SGD) optimizer and train the network for 12 epochs. We set the initial learning rate to 0.01 and decrease it by a factor of 0.1 after 8 and 11 epochs, respectively. We set the weight decay to 0.0001 and the momentum to 0.9. The batch size is 4, and the size of an input image is 768 × 1280. To train the network, we minimize the loss

$$L = L_{\text{cls}} + L_{\text{cen}} + L_{\text{coeff}}$$

where L_{cls} is the focal loss [8] between a predicted map P_i and the ground-truth $\bar{P}_i$, L_{cen} is the binary cross-entropy loss between a predicted map O_i and ground-truth $\bar{O}_i$, and L_{cls} is the polar IoU loss [12] between a predicted map R_i and the ground-truth $\bar{R}_i$.

Table 1. Comparison of AP_{50}, AP_{75}, and AP performances on the COCO2017 validation dataset.

	AP	AP_{50}	AP_{75}
PolarMask [12]	29.0	48.9	29.8
Proposed	**29.6**	**49.9**	**30.4**

Table 2. Comparison of AP_{50}, AP_{75}, and AP performances on the SBD validation dataset.

	AP	AP_{50}	AP_{75}
PolarMask [12]	27.7	50.6	25.7
Proposed	**30.7**	**54.9**	**29.6**

4.3. Comparative assessment

Quantitative comparison with PolarMask: We compare the instance segmentation results of the proposed network with those of PolarMask [12] on COCO2017 and SBD, by employing the same number M of parameters. In PolarMask, an object contour is described by M radial coordinates in a centroidal profile. On the other hand, in this work, the contour is represented by M eigencontour coefficients.

Table 1 shows the results on the COCO2017 validation dataset at $M = 36$. The average precision (AP) performance is measured, which averages AP scores across mask intersection-over-union (IoU) thresholds from 0.5 to 0.95 with an interval of 0.05. Also, we report AP_{50} and AP_{75} using thresholds 0.5 and 0.75, respectively. The proposed algorithm outperforms PolarMask in all three metrics. Table 2 compares the results on the SBD validation dataset. The proposed algorithm yields better results than PolarMask by significant margins in terms of all three metrics.

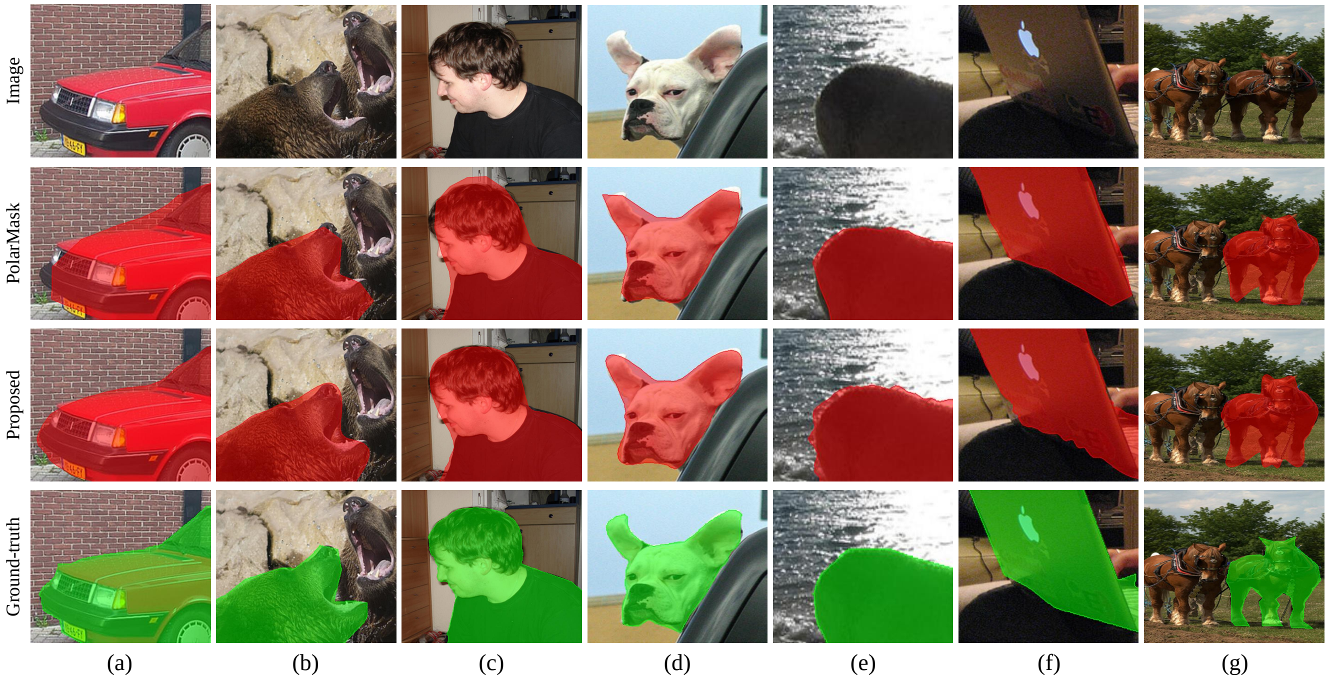

Figure 3. Qualitative comparison of instance segmentation results on the COCO2017 and SBD datasets.

The performance gaps between the proposed algorithm and PolarMask are marginal on COCO2017, whereas they are significant on SBD. For COCO2017, it is difficult to discover typical contour patterns due to the occlusions and the diversity of object shapes. Hence, eigencontour descriptors cannot represent the object boundaries properly. In contrast, for SBD, eigencontours describe object contours more reliably and thus more accurately, since the instances are classified into 20 categories only.

Qualitative comparison with PolarMask: Figure 3 compares the instance segmentation results qualitatively. In Figure 3 (a)∼(d), PolarMask fails to reconstruct complicated parts reliably, such as the bumper of a car or the animals' heads. In contrast, the proposed algorithm represents those boundaries more faithfully. Especially, the proposed algorithm reconstructs the curved parts of object boundaries more accurately. This is because eigencontours consist of a set of curved contours, which are used to represent object contours. However, this combination of eigencontours may create noisy signals, such as wiggling parts in Figure 3 (e) and (f). Also, both PolarMask and the proposed algorithm assume the star-convexity of object shapes, so they fail to preserve slender and hollow parts of objects, such as the horse legs in Figure 3 (g).

5. Conclusions

In this work, we applied eigencontour descriptors in [11] to the instance segmentation task. While the segmentation scheme in [11] is based on the ESE-Seg implementation, the proposed network is based on the PolarMask framework. Experimental results demonstrated that the proposed algorithm outperforms PolarMask on COCO2017 and SBD. Also, we analyzed the characteristics of eigencontour descriptors qualitatively.

References

[1] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science*. 2015. 2

[2] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *Proc. CVPR*, 2019. 1

[3] Edward R. Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Elsevier, 2004. 2

[4] Yuxin Fang, Shusheng Yang, Xinggang Wang, Yu Li, Chen Fang, Ying Shan, Bin Feng, and Wenyu Liu. Instances as queries. In *Proc. ICCV*, 2021. 1

[5] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *Proc. ICCV*, 2011. 1, 3

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. ICCV*, 2017. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 2

[8] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. CVPR*, 2017. 3

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Charles L. Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, 2014. 1, 3

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

[11] Wonhui Park, Dongkwon Jin, and Chang-Su Kim. Eigencontours: Novel contour descriptors based on low-rank approximation. In *Proc. CVPR*, 2022. 1, 2, 4

[12] 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 *Proc. CVPR*, 2020. 1, 2, 3

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