Parametric Skeleton Generation via Gaussian Mixture Models

Chang Liu[†], Dezhao Luo[†‡], Yifei Zhang[†‡], Wei Ke[§], Fang Wan[†] and Qixiang Ye[†*]

[†]University of Chinese Academy of Sciences, Beijing, China
[‡]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[§]Carnegie Mellon University, Pittsburgh, United States
{liuchang615,wanfang13}@mails.ucas.ac.cn, {luodezhao,zhangyifei0115}@iie.ac.cn
weik@andrew.cmu.edu, qxye@ucas.ac.cn

Abstract

We propose an efficient and effective control point extraction algorithm for parametric skeleton generation. The object skeleton pixels are predicted via an hourglass network and partitioned into skeleton branches using Gaussian Mixture Models. For each skeleton branch, a Bezier curve is utilized to generate the control points. The radius of the skeleton is computed by the distance between the border of the object and the Bezier curve. The branches are sorted by the area so that the parametric skeleton representation is unique. For the Parametric SkelNetOn competition, the proposed approach achieves the prediction score of 11793.89, which is in the first place on the performance leader-board.

1. Introduction

For the skeleton extraction task that draws attention for several decades, deep learning methods demonstrate unprecedented performance. Nevertheless, the problem about how to convert the discrete skeleton points to parametric representation remains seldom investigated.

Bezier curve is a spontaneous way for parametric skeleton, as shown in Fig. 1. The skeleton is fitted with Bezier curves and represented by the control points of the curves. By associating with the radius of each points, the object mask is well represented and reconstructed [1, 3, 7, 13].

Mestetsk *et al.* proposed several approaches for parametric representation [10, 11, 12]. In [10], a skeleton is represented as a plan view with edges that are linear and quadratic Bezier curves. A description of the radial function in the Bezier spline form is given. Mestetsk also proposed a approach using Compound Bezier Curves to represent the binary image skeleton [12]. After that, Mestetsk proposed a shape analysis of the continuous medial representation using Bezier curves [11].

[*]Corresponding author

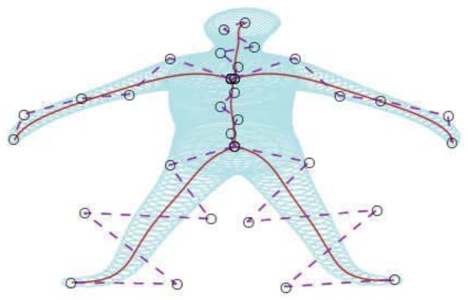
Figure 1. Illustration of a parametric skeleton. The skeleton is fitted with a Bezier curve and represented by the control points of the curve. By associating with the radius of points, the object mask is reconstructed.

In this paper, we utilize the Bezier curve to fit the discrete skeleton points and thereby creating an efficient and effective control point extraction algorithm for parametric skeleton generation. Given an image, a deep learning approach is used to predict skeleton curves of objects. The skeleton curve is then partitioned into skeleton branches by Gaussian Mixture Models (GMMs). For each skeleton branch, Bezier curve fitting is utilized to generate the control points. The radius of the skeleton is computed by the distance between the border of the object and the curve. The branches are sorted by the area of branches so that it reduces discontinuous fragments on the skeleton curve.

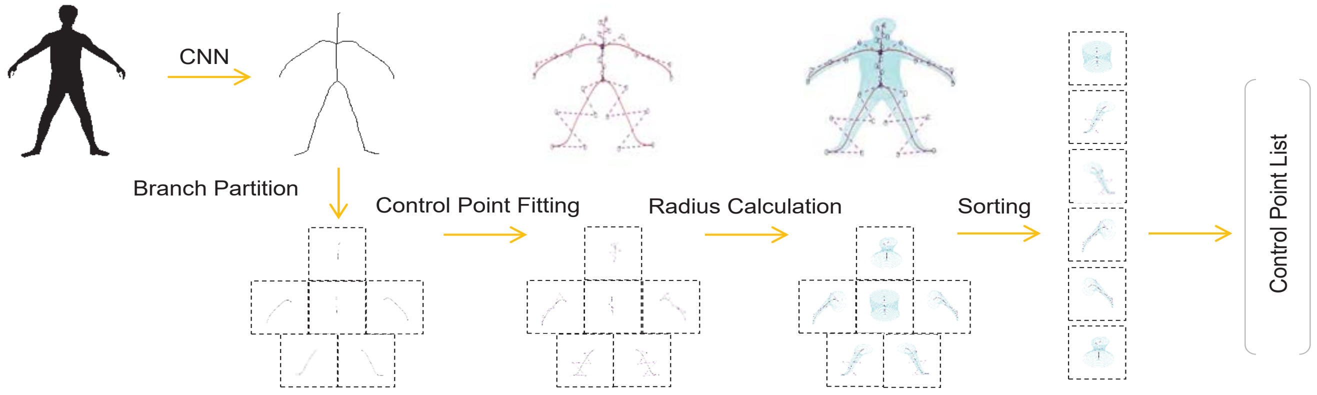

Figure 2. Flowchart of the proposed parametric skeleton generation procedure. (Best viewed in color and zoom in.)

2. Methodology

Supposing an image I has skeleton S, the parametric skeletons is represented as $\{C_k, r_k\}_{k=1}^K$, where C_k and r_k are control points and radius for the k-th skeleton branch, and K is the number of skeleton branches. The parametric skeleton generation approach is shown in Fig. 2 and detailed in Algorithm 1, which consists CNN-based skeleton predicting, GMMs-based skeleton branch producing, Bezier curve fitting, radius calculating, and area based sorting.

Skeleton prediction. We build a fully convolutional neural network (FCN) for skeleton prediction, by updating the VGGNet to an hourglass architecture with the purpose of increasing the size of receptive field, named feature hourglass network (FHN) [5]. By adding a 1×1 convolutional layer to each convolutional layer of the hourglass, side-outputs are generated. The final skeleton S is by fusing the side-outputs.

Branch partition. Following the second step of Algorithm 1, a skeleton is divided to branches by GMMs.

Control point fitting. Given the control points $\{C_k\}$, a Bezier curve is constructed as

$$B_k(t) = \sum_{j=0}^{n} \binom{n}{j} P_j (N-t)^{n-j} t^j, t \in [0, N], \quad (1)$$

where $P_j = C_{k,j}$ denotes the j-th control point. We set $n = 5$ as the 6^{th} order Bezier curves are used.

To fit the skeleton, we design an optimization procedure to calculate the control points as Algorithm 1. Six control points are randomly chosen for initialization. A Bezier curve is constructed with these control points using Eq. 1. The control points are updated by minimizing the objective loss function as

$$\mathcal{L}_{k,t} = \sum_{t=0}^{N} ||(B_k(t) - L_{k,t}||_2. \quad (2)$$

Algorithm 1: Parametric Skeleton Generation.

```
Input: An image $I$
Output: Parametric skeletons $\{C_k, r_k\}_{k=1}^K$, which is
        represented by the control points and radius
        of $K$ skeleton branches.
1. Predict skeleton $S$ of $I$ with a FCN.
2. Produce branches $\{L_k\}_{k=1}^K$ via GMMs.
   2.1 Set threshold $\theta$ for loss computed by GMMs.
   2.2 Set default branch number $K$.
   2.3 Cluster skeleton pixels $S$ into $K$ groups via
  GMMs. $K$ groups refer to $K$ branches $\{L_k\}_{k=1}^K$.
   2.4 Calculate the loss of GMMs.
   2.5 Increase $K$ and go back to 2.3, if loss $\geq \theta$
3. Fit the branches $\{L_k\}_{k=1}^K$
for $k \leftarrow 1$ to $K$ do
      3.1 Calculate length $N$ for $k$-th branch $L_k$.
      3.2 Initialize $n+1$ points $C_k$ as control points
     for the branch.
      3.3 Calculate $N$ points of Bezier curves $B(t)$
     with $C_k$ as Eq. 1.
      3.4 Compute the loss between the Bezier curves
     $\{B_k(t)\}_{t=0}^N$ and branch $\{L_{k,t}\}_{t=0}^N$ with Eq. 2.
      3.5 Go to 3.3 if loss $\leq 0.0001$.
end
4. Calculate the distance ($r$) between control points
  $C_{k,j}$ and a projected point $C_{k,j}^p$ in the border. $r$ is
  computed as Eq. 3 $C_{k,j}$ refers to the $j$-th control
  point of $k$-th branch.
5. Sort all branches by their areas calculated by Eq. 4.
```

Radius calculation. The final Bezier curve is constructed with the optimized control points. Each control point, $C_{k,j} = (x_{k,j}, y_{k,j})$, is projected to the Bezier curve to find a projected point $C_{k,j}^p = (x_j^p, y_j^p)$. The radius is calculated as the minimum distance between the projected point and the border of the object mask, as

$$r_{k,j} = \begin{cases} \quad ||C_{k,j} - C^p_{k,j}||_2 & B^l \in Shape \\ - \quad ||C_{k,j} - C^p_{k,j}||_2 & B^l \notin Shape. \end{cases} \tag{3}$$

Sorting. Sorting is used to output a single parametric skeleton without ambiguity. The skeleton branches are sorted by their part areas calculated as

$$S_k = \frac{1}{2}\sum_{j=0}^{n-1}(r_{k,j} + r_{k,j+1}) * ||C_{k,j+1} - C_{k,j+1}||_2. \tag{4}$$

3. Experimental results

In this section, we describe the experimental settings, approah implementation details, and the parametric skeletion generation performance.

3.1. Dataset and Performance Metrics

Dataset: Parametric SkelNetOn contains 1,725 shape images, where 1,219 images are used for training, 242 for validation, and 266 for testing. The parametric skeleton ground-truth is given as a set of triplets like $\{x_{j,k}, y_{j,k}, r_{j,k}\}_{k=1}^K$, where $j = 1 \dots 6$ denoting 6 control points, K is the number of skeleton fragments. x, y and r respectively denote the coordinates of the control point and the radius of skeleton points.

Protocol: The parametric skeleton generation results are evaluated by the mean squared distance (MSD) between the control points on the original and predicted branches, and the missing branch error (MBE) for each missing or extra branch b. The MSD is defined as

$$MSD(b, \tilde{b}) = \frac{1}{6}\sum_{i=0}^{5}\left((x_i - \tilde{x_i})^2 + (y_i - \tilde{y_i})^2 + (r_i - \tilde{r_i})^2\right),$$

where $b = (x_i, y_i, r_i)_{i=\{0..5\}}$ is a branch in the ground truth, and $\tilde{b} = (\tilde{x_i}, \tilde{y_i}, \tilde{r_i})_{i=\{0..5\}}$ is the corresponding branch in the output. The MBE is defined as

$$MBE(b) = \frac{1}{5}\sum_{i=0}^{4}\left((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2\right) + \frac{1}{6}\sum_{i=0}^{5} r_i^2.$$

The evaluation function between an output vector $\tilde{V}$ and its ground truth V is defined as

$$D(V, \tilde{V}) = \frac{1}{N_b}\sum_{j=0}^{n_b-1} MSD(b^j, \tilde{b^j}) + \sum_{j=n_b}^{N_b-1} MBE(\hat{b^j}),$$

where N_b and n_b respectively denote the minimal and maximal numbers of branches in the ground truth and in the output, and $\hat{b}$ are branches in the vector containing the maximal number of branches.

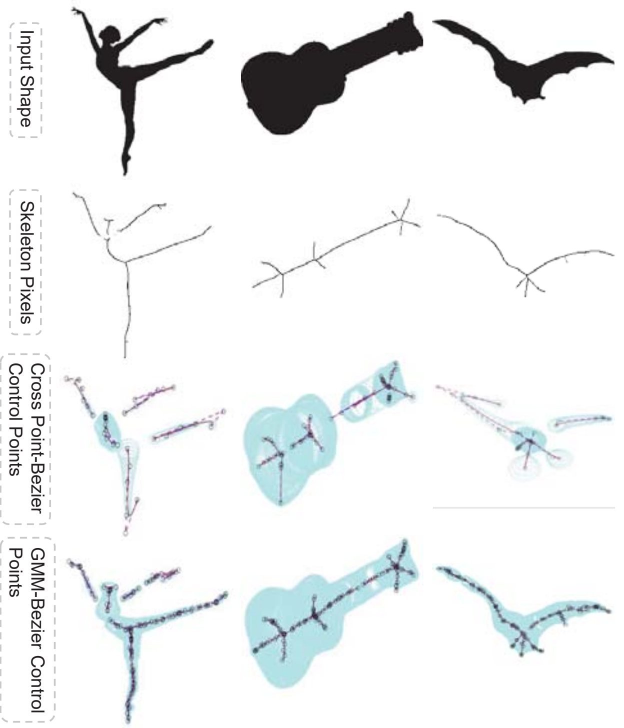

Figure 3. Examples of parametric skeleton results on the Parametric SkelNetOn test set.

3.2. Skeleton Pixel Prediction

Network architecture: We built the fully convolutional neural network for skeleton pixels prediction with hourglass architecture, which is built on the VGGNet [14]. The first four stages of VGG are kept, while the pool4 is removed. Then we insert deconvolutional layers among the convolutional layers of the fifth stage. In addition, the dilation technique [2] is applied in the convolutional layers to further enlarge the receptive field and facilitate extracting fine-details. Rich side-outputs are generated by adding a 1×1 convolutional layer to each layer in the hourglass backbone and hierarchical side-outputs are fused [8, 9].

Data augmentation: We follow [6] to rotate each image to 4 angles (0°, 90°, 180°, 270°) and flip it with three axes. Finally, we re-size each training image to 3 different scales (0.5, 1, 1.5).

Model parameters: Following the setting of RSRN [8], we train the network by fine-tuning the pre-trained 16-layer VGGNet [14]. The hyper-parameters include: mini-batch size (1), learning rate (1e-6 for Pixel SkelNetOn), loss-weight for each RU output (1.0), momentum (0.9), weight decay (0.0002), and maximum number of training iterations (150k). In the testing phase, a standard non-maximal suppression algorithm [4] is applied on the output map.

Backbone		Backbone with side-outputs		
VGG	H-VGG	H-RCF	H-HIFI	H-RSRN
12.71	50.39	59.12*	64.56*	**64.77***

Table 1. F-measure on the Pixel SkelNetOn validation set. * indicates F-measure on the test set. "H-" denotes that the feature hourglass network (FHN) [5] backbone is used.

Skeleton pixel prediction: The evaluation of skeleton extraction is shown in Table 1, which shows that the hourglass architecture significantly outperforms the VGG backbone. With hourglass architecture and the hierarchical integration strategy, H-RSRN achieves the best F-measure of 64.77% on the test set.

3.3. Parametric Skeleton Generation

3.3.1 Skeleton Branch Partition

Cross points and **GMMs** are respectively used to partition skeleton branches given skeleton pixels. A cross point is identified if there are more than 2 of the 8 skeleton pixels around it. Deleting the cross points can divide a skeleton to branches using connected component analysis. For the loss of GMMs varies greatly in the dataset, it is hard to use a fixed threshold to get the accurate number of branches of different shape. Instead, we fix the number of branches to 20 as a matter of experience.

The comparison of the branch extraction approaches is shown in Table 2. It can be seen that GMMs significantly outperforms cross-points approach. We observed that a long skeleton branch could be broken by a wrong cross point. In contrast, GMMs can cluster separate points in the same branches.

3.3.2 Skeleton Branch Sorting

We propose using max area and max radius to sort skeleton branches.

Max area: We construct 5 trapezoids with six control points for each branch. Branches are sorted by the total area of trapezoids.

Max radius: Branches are sorted by the max radius of each branch. In Table 2, one can see that sorting the parametric skeleton by the max radius of each skeleton branch achieves better performance than by the area of trapezoids.

3.3.3 Skeleton Branch Fitting

We do not implement fitting branches via a settle threshold as the loss varies greatly. We instead try setting different iterations in training. In Table 3, we compared the performance with different training iteration numbers. One can see that the score of 50 times is better than those of 25 and 100 either in radius-based sorting or area-based sorting.

The final performance on validation set and test set is shown in Table 4, which is from the leader-board of the Parametric SkelNetOn challenge [3].

	Cross points	GMMs	Cross points	GMMs
Sorting	Max area	Max area	Max radius	Max radius
Score	13366.37	12335.72	13383.23	**11796.58**

Table 2. Comparisons of branch partition and sorting strategies. Lower score indicates better performance.

Sorting Method	Iteration	Score
Max radius	25	11933.16
	50	**11793.89**
	100	11796.58
Max area	25	12433.02
	50	12376.80
	100	12335.72

Table 3. Performance of different hyperparameters, with the setting of GMMs based skeleton branch generation.

Team	Prisdl	x2	RG
Val	**10609.96**	20532.78	12705.34
Test	**11793.89**	12462.39	13277.47

Table 4. Performance comparison on validation and test datasets of the Parametric SkelNetOn Challenge.

4. Conclusion

In this paper we propose a parametric skeleton generation approach. The skeleton is predicted by hourglass convolutinal neural network and partitioned into skeleton branches via Gaussian Mixture Models (GMMs). The Bezier curve is utilized to parametrize the skeleton branches and parametric skeletons are sorted according to the area of branches. Experimental results show that the proposed parametric skeleton generation approach achieves the prediction score of 11793.89 on test dataset, which is in the first place on the performance leader-board.

Acknowledgement

The authors are very grateful to the support from NSFC grant 61836012 and 61671427, and Beijing Municipal Science and Technology Commission grant Z181100008918014.

References

[1] Alexander M. Bronstein, Michael M. Bronstein, Alfred M. Bruckstein, and Ron Kimmel. Analysis of two-dimensional non-rigid shapes. *IJCV*, 2008.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE PAMI*, 40(4):834–848, 2018.

[3] Ilke Demir, Camilla Hahn, Kathryn Leonard, Geraldine Morin, Dana Rahbani, Athina Panotopoulou, Amelie Fondevilla, Elena Balashova, Bastien Durix, and Adam Kortylewski. SkelNetOn 2019 Dataset and Challenge on Deep Learning for Geometric Shape Understanding. *arXiv e-prints*, 2019.

[4] Piotr Dollár and C. Lawrence Zitnick. Fast edge detection using structured forests. *IEEE PAMI*, 37(8):1558–1570, 2015.

[5] Nan Jiang, Yifei Zhang, Dezhao Luo, Chang Liu, Yu Zhou, and Zhenjun Han. Feature hourglass network for skeleton detection. In *IEEE CVPR Workshop*, 2019.

[6] Wei Ke, Jie Chen, Jianbin Jiao, Guoying Zhao, and Qixiang Ye. SRN: side-output residual network for object symmetry detection in the wild. In *IEEE CVPR*, pages 1068–1076, 2017.

[7] Kathryn Leonard, Geraldine Morin, Stefanie Hahmann, and Axel Carlier. A 2d shape structure for decomposition and part similarity. In *ICPR*, pages 3216–3221, 2016.

[8] Chang Liu, Wei Ke, Jianbin Jiao, and Qixiang Ye. Rsrn: Rich side-output residual network for medial axis detection. In *IEEE ICCV Workshop*, pages 1739–1743, 2017.

[9] Chang Liu, Wei Ke, Fei Qin, and Qixiang Ye. Linear span network for object skeleton detection. In *ECCV*, pages 136–151, 2018.

[10] Leonid Mesteskiy. Skeleton representation based on compound bezier curves. In *VISAPP*, 2010.

[11] Leonid Mesteskiy and B. H. Shekar. Bezier curve based continuous medial representation for shape analysis: A theoretical framework. In *MIKE*, 2017.

[12] Leonid Mestetskiy. *Binary Image Skeleton Representation by Compound Bezier Curves*. 2013.

[13] Thomas B Sebastian, Philip N Klein, and Benjamin B Kimia. Recognition of shapes by editing their shock graphs. *IEEE PAMI*, 26(5):550–571, 2004.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.