Multimedia Tools and Applications (2020) 79:1585–1602
https://doi.org/10.1007/s11042-019-08158-z

2D freehand sketch labeling using CNN and CRF

Xianyi Zhu[1] · **Yi Xiao**[1] · **Yan Zheng**[2]

Received: 24 September 2018 / Revised: 20 June 2019 / Accepted: 2 September 2019 /
Published online: 5 November 2019

Abstract
Accurate and fast sketch segmentation and labeling is a hard task, since sketches have much fewer features than natural images. This paper proposes a novel hybrid approach for fast automatic sketch labeling, which is based on convolutional neural network (CNN) and conditional random field (CRF). Firstly, we design a CNN for stroke classification. The CNN is equipped with larger first layer filters and larger pooling, which is suitable for extracting descriptive features from strokes. Secondly, we integrate each stroke with its host sketch to construct a more informative input for the CNN model. Finally, we leverage the spatio-temporal relations among strokes in the same sketch to create a connected graph, based on which we apply a CRF model to further refine the result of the CNN. We evaluate our method on two public benchmark datasets. Experimental results demonstrate that our method achieves the state-of-the-art level on both accuracy and runtime.

Keywords 2D sketch labeling · Stroke classification · Convolutional neural network · Conditional random field · Connected graph creation

1 Introduction

Freehand sketching is an effective and intuitive manner for people to present their thoughts. With the popularity of touchscreen devices, sketching becomes easy for humans. Many studies and applications on sketch have been investigated in recent years, including sketch recognition [6, 32, 47], sketch-based image retrieval and synthesis [7, 18, 28, 29, 36, 38, 50], sketch-based 3D model retrieval [14, 15]. However, most researches on sketch understanding are at object level. Accurate and automatic semantic understanding of sketch at part level is also required, which helps to reduce manual segmentation operations in applications, such as sketch-based 3D modeling [3, 44, 45].

In general, each sketch in existing datasets [2, 8, 31] is composed of a sequence of ordered strokes. Translating the partial semantics of sketches is a quite challenging task.

✉ Yi Xiao
yixiao_csee@hnu.edu.cn

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha, People's Republic of China

[2] College of Electrical and Information Engineering, Hunan University, Changsha, People's Republic of China

Sketches provide few visual cues: they consist of black and white lines without any color. Sketches are highly abstract and diversified: complex entities can be depicted with a few strokes, and the same entity can be drawn in multiple styles. Strokes in a sketch have varied levels of deformation: strokes with the same semantics may have different length and shape, and vice versa.

The task of this paper is to assign each stroke of a sketch from a given category with a label. The label comes from a pre-defined set of labels. Huang et al. [8] first addressed an interactive solution and achieved impressive results by constructing correspondences between the sketch and already labeled 3D models. Then, Schneider and Tuytelaars [31] first proposed an automatic technique and achieved comparable results by adapting a conditional random field (CRF) model to classify strokes. Recently, Li et al. [16] proposed to acquire a large number of training data by rendering already labeled 3D models for their network training. The 3D data-driven methods [8, 16] may not be feasible for the abstract sketches that have no corresponding 3D meshes, and the hand-crafted features (dense SIFT [21] and spatial coordinates) based method [31] is sensitive to huge variations of the sketches. In contrast, our method needs no 3D models, and the descriptive features are based on learning.

Accurate and fast automatic labeling depends on three key factors. (i) **Fast and descriptive feature extraction for strokes.** To accurately differentiate various strokes in realtime, we need to extract descriptive features for strokes. As convolutional neural networks [10, 12, 34] (CNNs) have been shown its ability to learn effective feature from raw inputs, and run fast in the feed-forward phase, we design a CNN for stroke classification. The CNN has larger first layer filters and larger pooling, which helps to extract descriptive features from stokes. (ii) **The host sketch.** The labels of strokes in a sketch are related to the host sketch that they form. Similar strokes in different host sketches probably have different semantic labels. We integrate each stroke and its host sketch as an auxiliary informative input to the CNN. (iii) **The relations among strokes.** Strokes in the same sketch also relate to each other. However, in order to utilize relations on spatial, large computation is needed to discover high-level spatial relations (enclosure in [31]) among strokes. Therefore, we propose to combine the temporal relations (order) with low-level spatial relations (intersection) among strokes to create a connected graph. Then, we use a CRF model to find a global optimal configuration. Extensive experimental results show that our technical elements are effective.

In summary, the main contributions consist of three aspects. Firstly, we present a hybrid method based on CNN and CRF for 2D sketch labeling. Secondly, we propose to augment the input of CNN by integrating the stroke and its host sketch. Finally, we utilize the spatio-temporal relations among strokes to build a connected graph, based on which we apply a CRF model to further improve the result of the CNN.

The rest of this paper is structured as follows. Section 2 introduces the related works about sketch segmentation. Section 3 describes the technical elements of our method. Section 4 provides extensive experimental results. Section 5 discusses limitations and future research directions. Section 6 concludes this paper.

2 Related works

The three studies [8, 16, 31] are highly related to ours. Huang et al. [8] adopted a mixed integer programming formulation to transfer part segmentations and labels to an object sketch from a 3D model database. However, their method can not directly process the abstract sketches which have no corresponding 3D models, and their method perhaps needs manual

viewpoint tuning to align sketches with 3D models accurately. They reported that their method needs 40 minutes to segment a sketch. It is not appropriate for real-time applications. To avoid these limitations, Schneider and Tuytelaars [31] adapted a CRF model to automatic segment sketches without manual interaction. Their unary classifier uses the handcrafted features (dense SIFT [21] and spatial coordinates) encoded by Fisher Vectors [27]. The handcrafted feature has a limited descriptive ability, which may be sensitive to various strokes. They proposed to exploit spatial relations (intersection and enclosure) among strokes to create a graph. However, their graph creation is time-consuming and they can not ensure the graph is connected. In contrast, the feature and graph creation in our method are different from theirs. We use the feature learned by CNN, and we leverage the spatio-temporal relations among strokes to fast create a connected graph. Recently, Li et al. [16] combined a fully convolutional network (FCN) with a graph-cut model for fast sketch segmentation and labeling. Their training data are obtained by rendering corresponding 3D models with labels so that their method can not work on abstract sketches without labeled 3D models. Their network and graph model are different from ours. Specifically, our network mainly consists of convolution and fully connection layers, while their network mainly consists of convolution and transposed convolution layers. In our graph model, each node represents a stroke. In their graph model, each node represents a pixel (point).

Many studies are also related to ours. Noris et al. [23] presented a scribble-based interface for user-guided segmentation of digital sketchy drawings. They introduced a novel energy minimization formulation in which both geometric and temporal information is used. Their work focuses on interactive selecting strokes. Sun et al. [35] proposed a sketch segmentation framework based on graph algorithm to segment objects from sketch scenes. They adopted large clip-art images as the prior knowledge to merge strokes that belong to the same objects. Their task is to perform sketch instance segmentation at object level in sketch scenes. Qi et al. [24, 25] adopted perceptual grouping principles to group strokes of a sketch into semantic parts. They computed an affinity matrix between strokes using hand-crafted features based on proximity and continuity principles. Their task is to predict whether strokes belong to the same group rather than what group. Sarvadevabhatla et al. [30] proposed a two-level fully convolutional network for fully automatic parsing of freehand object sketches. Their method labels object regions as parts, rather than all strokes. The object region boundary may include non-stroke pixels. Kim et al. [9] addressed instance segmentation at stroke level for rasterized sketch vectorization. They adopted a pair of neural networks to predict the probability that pairwise pixels belong to the same stroke and the probability that a pixel belongs to overlap regions, and then combined the probabilities by solving a Markov random field [13]. Some recent studies are related to our CNN and integration of sketch and stroke, such as alcoholism identification [43], tea category classification [48], image super-resolution [49], image captioning [39, 42], learn to recommend [41], multimodal representation learning [20, 40], and attribute extraction [17, 19].

3 Our method

In this section, we introduce the three key elements of our method. We first describe our basic CNN architecture and explain its specific aspects (Section 3.1). We then introduce integration of stroke and sketch which makes the input more meaningful (Section 3.2). We next introduce how to build the graph based on spatio-temporal relations among strokes, and how to apply the CRF model to predict the globally optimal configuration (Section 3.3). Figure 1 shows the pipeline of our method.

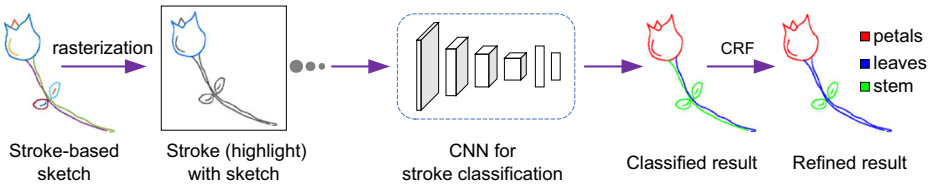


Fig. 1 The pipeline of our overall method

3.1 A CNN for stroke classification

Since a stroke is a more abstract and sparse sketch, we configure the CNN for stroke classification inspired by Sketch-a-Net [47], a state-of-the-art CNN for sketch object classification ($\approx$ 80% accuracy on benchmark TU-Berlin dataset [2]). The architecture of our CNN is as follows. The input's height and width are 300. Then, the CNN begins with four convolutional blocks, in which each block consists of one convolutional (Conv) layer followed by a rectified linear unit (ReLU) and a max pooling (Max Pool) layer. The output size of the fourth convolutional block is 3×3 rather than 1×1, in order to maintain enough information to represent the input. Next, two fully connected layers are appended to the fourth convolutional block. Dropout regularization is applied to the first fully connected layer for reducing overfitting. The final layer has *nLabel* output units corresponding to the n labels. The loss function is the softmax loss. The details of our CNN are illustrated in Fig. 2.

Our CNN has the following four main characteristics:

(i) **Larger first layer filters.** The first layer filters play an important role, because all subsequent computations are dependent on their output. The state-of-the-art networks [5, 34] tends to put small-size filters (3×3 or 7×7) in the first layer for natural image tasks. However, Sketch-a-Net shows that larger filters are more helpful for sketch modelling, as sketches have no texture information [47]. For example, a small patch within a circle shape can be regarded as mouth or eye based on its texture, but this is not practicable to sketches. Therefore, we employ filters with size 15×15 in the first convolutional layer, in order to extract more structured context.

(ii) **Larger stride for the first layer filters.** The stride of the first convolutional layer in the Sketch-a-Net is set to 3. As strokes are sparser than sketches, larger stride helps capture more structured context. We set the stride to 4 in our first layer.

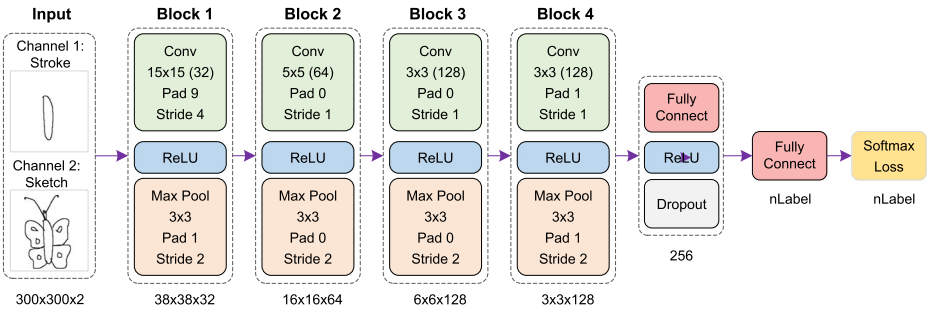


Fig. 2 The structure of the CNN for stroke classification

(iii) **Larger pooling size.** The 2×2 max pooling with stride 2 is used in photo-oriented CNN [34] for keeping some spatial invariance in downsampling. However, we found that the overlapping 3×3 pooling with stride 2 could bring a 4% improvement for sketch-oriented CNN in the experiment.

(iv) **Larger feature map.** The Sketch-a-Net has six convolutional layers. The size of the feature map generated by the sixth convolutional layer is 1×1 with 512 channels. We argue that a slightly larger feature map can keep more information. Thus, the size of the feature map of our last convolutional layer is 3×3 with 128 channels.

3.2 Integration of stroke and sketch

Individual strokes only can provide limited information. In fact, the label of a stroke is highly related to the host sketch. For example, even identical strokes are likely to own different semantic labels if they are putted in different host sketches. Therefore, we propose to use the host sketch as the reference object to enhance the strokes. Figure 3 compares two groups of examples that consider the host sketch as reference objects or not. It is obvious that referencing the host sketch makes people recognize a stroke easily. Even though we use rotation for data augmentation, a common measure to reduce overfitting when training, strokes' position and orientation are still clear.

We implement the idea through a simple and effective approach, extending our single-channel CNN to a two-channel CNN: the input size is changed to $300 \times 300 \times 2$ in which the first channel is the stroke and the second channel is the host sketch. The filter size in the first convolutional layer is changed to $15 \times 15 \times 2$ accordingly. This modification makes data augmentation effective, and the number of parameters and training time are increased by less than 1%.

3.3 Refinement with CRF

In the former two subsections, the results outputted by the CNN model already reaches a state-of-the-art level (see Section 4.1), but the CNN model ignores the relations among

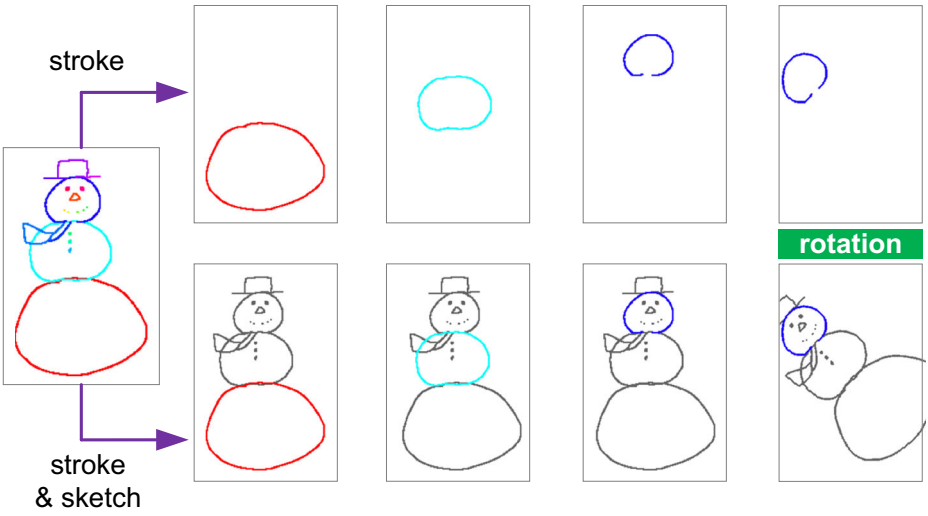


Fig. 3 Two groups of examples adopt the host sketch as reference object or not

strokes. An optimization on these relations is necessary, which helps to find a globally optimal configuration for a sketch. We choose a CRF model to formulate this optimization. The CRF [11] has been proven to be a good structure for probability description in a graph.

In order to use CRFs for a sketch, we have to define a connected graph $G = (V, E)$, where the V is the set of strokes on a sketch, and the E is the set of relations among strokes. Let i represent a stroke, s_i be its descriptive feature related to node, N_i be its neighboring strokes in a graph, and e_{ij} be the descriptive feature related to the edge between nodes (strokes) i and j. The labels $\{l_i | i \in V\}$ of these strokes can be optimized by solving this objective function:

$$\max_{\{l_i, i \in V\}} \frac{1}{Z} \prod_{i \in V} \phi_u(l_i | s_i) \prod_{i \in V, j \in N_i} \psi_p(l_i, l_j | e_{ij}) \tag{1}$$

where ϕ_u is unary node potential function, ψ_p is pairwise edge potential function between two connected nodes, and Z is a normalization factor to ensure that the probabilities range from 0 to 1.

Graph creation It is quite challenging to define which strokes should be connected. If we only connect the crossed strokes, we do not guarantee the graph is connected. If we also connect strokes to those strokes that are enclosed by them, it is time-consuming to compute (discover) the enclosure relations. Instead of exploiting other new spatial relations among strokes, we introduce an inherent characteristic of stroke-based sketch, the temporal relation. We first connect the strokes according to their stroke order to obtain a connected graph. Then, we connect those crossed strokes. The graph is meaningful, because strokes that are adjacent on temporal order or on space trend to related semantics.

In summary, the edges of our graph can be divide into three classes: spatial, temporal, and spatio-temporal. We show some subgraph examples in Fig. 4. For the face sketch, though its all strokes are not crossed, we can create a connected graph by connecting the strokes that are adjacent on temporal order.

Potential function definition The unary potential $\phi_u(l_i | s_i)$ is defined as the CNN model output, i.e., the probability that stroke i has the label l_i. The feature s_i is actually the feature map learned by the CNN model.

For the pairwise potential $\psi_p(l_i, l_j | e_{ij})$, the feature e_{ij} is the type of edge: spatial, temporal, and spatio-temporal. We calculate the probabilities about that edges connect labels. The probabilities can be calculated by counting on the training data. Therefore, we define the potentials $\psi_p(l_i, l_j | e_{ij})$ as the corresponding probabilities. Specifically, for each type of relation, we define an $m \times m$ matrix P, where m is the number of possible labels. The matrix P has two kinds of values: the probability that an edge connects two nodes which have the same label, in the diagonal, and the probability that an edge connects two nodes

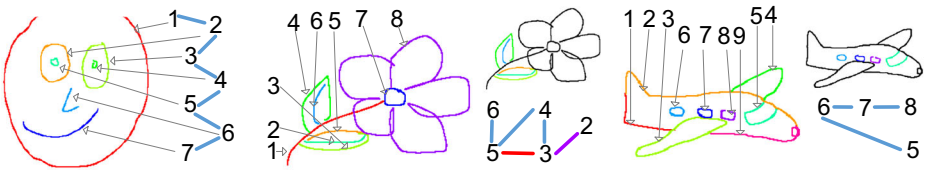


Fig. 4 Three sketches that each shaded stroke is marked with corresponding stroke order, and six examples of subgraphs. Spatial edges are in red, temporal edges are in blue, and spatio-temporal edges are in purple

with different labels otherwise. The potentials $\psi_p(l_i, l_j|e_{ij})$ are the elements of the corresponding P. Then, we use iterated conditional mode [1] to solve (1) to find the globally optimal configuration. We use the MATLAB toolbox UGM [22] to implement this solving.

4 Experimental results

Datasets We evaluate our method on two benchmarks: Huang dataset [8] and Schneider dataset [31] (a subset of TU-Berlin dataset [2]). Huang dataset consists of 10 object categories and each category has 30 sketches, while Schneider dataset consists of 6 object categories and each category has 20 sketches. The two datasets represent two different types of sketches respectively. The sketches in Huang dataset resemble real-world 3D objects somewhat, while the sketches in Schneider dataset are more abstract.

Metrics Like related works [8, 16, 31], two metrics are used to evaluate sketch segmentation and labeling: *pixel metric*, the percentage of pixels that are assigned with correct labels in a sketch; *component metric*, the percentage of strokes that are assigned with correct labels in a sketch. A stroke is correctly labeled if the percentage of pixels correctly labeled in the stroke is over a given threshold (75% in the experiments). The pixel metric has a bias towards large components that have more pixels, and the component metric gives a bias to small components that are of larger number.

Implementation Our experimental platform is a PC with an Intel Core I7-4790k 4.0 GHz CPU and 16GB RAM. We train our CNN model on an Nvidia GTX1080ti GPU with 11GB memory. We use Tensorflow framework with Python API to implement the CNN model. To reduce overfitting, we apply data augmentation by rotation and horizontal mirror. Specifically, we randomly rotate strokes by $-15 \sim 15$ degrees, and flip strokes horizontally. We simply use the common reflection transformation manners to conduct data augmentation, because other image processing manners, such as color jittering and contrast, may not work on binary sketches. We choose the mini-batch stochastic gradient descent to train the CNN model. The dropout rate is 70%. The number of iteration is 1100. The learning ratio is 0.001, and the decay ratio is 0.5 per 500 iterations. The batch size is 50.

4.1 Evaluation on Huang dataset

Accuracy We perform 3-fold cross validation (3FCV) and 5-fold cross validation (5FCV) experiments on Huang dataset. Specifically, 3FCV splits the 30 sketches in each category into 3 equal subsets, and each subset contains 10 sketches. In each run, a single subset is used for testing and the other two subsets (20 sketches) are used for training. This is repeated for all 3 folds. 5FCV splits each category into 5 equal subsets. Four subsets (24 sketches) are used for training and the rest one subset is used for testing in each run, and this is repeated for all 5 folds.

The sample splitting of 3FCV is consistent with Schneider et al. [31]'s splitting. However, Li et al. [16]'s training data are acquired from additional labeled 3D meshes. The labels of 3D meshes are the same as those of sketches, and the number of 3D meshes is larger than 20. In other words, their training size is larger than that of 3FCV. To make a fair comparison, we compare our results yielded by 3FCV with Schneider et al. [31] and Huang et al. [8]'s results, and we compare our results yielded by 5FCV with Li et al. [16]'s results.

Table 1 Pixel accuracy results on Huang dataset

Method	K-fold	Airplane	Bicycle	Cdlbrm	Chair	Fourleg	Human	Lamp	Rifle	Table	Vase	Average
Huang-Manual [8]	–	82.4	78.2	72.7	76.5	80.2	79.1	92.1	75.9	79.1	71.9	78.8
Huang-Auto [8]	–	74.0	72.6	59.0	52.6	77.9	62.5	82.5	66.9	67.9	63.2	67.9
Schneider-CRF [31]	3–fold	55.1	79.7	72.0	66.5	81.5	69.7	82.9	67.8	74.5	83.3	73.3
Ours-C	3–fold	79.4	80.9	71.4	66.3	83.1	76.7	93.5	63.1	79.5	**89.0**	78.3
Ours-CS	3–fold	81.1	84.0	**72.2**	70.7	84.6	78.6	95.0	68.3	**82.3**	86.9	80.4
Ours-full-3	3–fold	**81.4**	**84.3**	72.1	**71.9**	**84.7**	**78.9**	**95.0**	**68.8**	82.1	88.2	**80.8**
Li-FCN [16]	–	**85.7**	86.2	78.3	**75.5**	84.9	**80.5**	87.8	70.6	81.4	82.7	81.4
Ours-full-5	5–fold	82.5	**88.1**	**79.1**	71.1	**85.7**	79.4	**96.3**	**71.1**	**85.7**	**89.1**	**82.8**

The Huang-Manual and Huang-Auto are the manual and automatic versions of Huang et al. [8]'s method respectively

Tables 1 and 2 compare the results using pixel metric and component metric accuracies respectively. Best automatic results are in bold. Both Huang et al. [8] and Schneider et al. [31]'s methods should split up a complete stroke into segments at 'X' and 'T' junction or high-curvature points. Their component accuracies are the percentage of segments correctly labeled in a sketch. However, their stroke splitting points may be different. Therefore, the pixel metric comparison is fairer.

See Table 1, our single CNN model (Ours-C) already outperforms Huang-Auto, Schneider-CRF. And the CNN model is comparable to Huang-Manual. With the integration of stroke and sketch, our average accuracy (Ours-CS) improves 2.1%. And with CRF refinement, our average accuracy (Ours-full-3) increases by 0.4%. This improvement demonstrates that each individual component of our method is effective. Our full method outperforms Schneider-CRF by 7% and Huang-Manual by 2% on average. Compared with Li-FCN [16] in Table 1, our full method (Ours-full-5) wins in seven categories and outperforms 1.4% on average. For the categories *lamp* and *vase*, our method outperforms Li-FCN by high percentages, e.g., 8.5% and 6.4% respectively.

Table 2 Component accuracy results on Huang dataset

Method	K-fold	Airplane	Bicycle	Cdlbrm	Chair	Fourleg	Human	Lamp	Rifle	Table	Vase	Average
Huang-Manual [8]	–	66.2	66.4	56.7	63.1	67.2	64.0	89.3	62.2	69.0	63.1	66.7
Huang-Auto [8]	–	55.8	58.3	47.1	42.4	64.4	47.2	77.6	51.5	56.7	51.8	55.3
Schneider-CRF [31]	3–fold	48.7	68.6	66.2	61.6	74.2	63.1	77.2	**65.1**	65.6	79.1	66.9
Ours-C	3–fold	75.5	75.1	68.2	60.3	77.6	70.8	90.5	59.1	73.3	**86.0**	73.6
Ours-CS	3–fold	**77.1**	78.7	69.6	63.8	79.3	72.1	95.1	64.5	76.7	84.2	76.1
Ours-full-3	3–fold	76.1	**79.1**	**70.0**	**64.4**	**79.5**	**73.0**	**95.1**	64.6	**76.9**	84.8	**76.4**
Li-FCN [16]	–	76.9	77.1	69.9	**70.3**	75.5	72.8	83.8	65.7	77.3	78.0	74.7
Ours-full-5	5–fold	**77.0**	**82.1**	**74.2**	63.5	**81.0**	**73.3**	**95.0**	**66.5**	**80.5**	**85.8**	**77.9**

The Huang-Manual and Huang-Auto are the manual and automatic versions of Huang et al. [8]'s method respectively

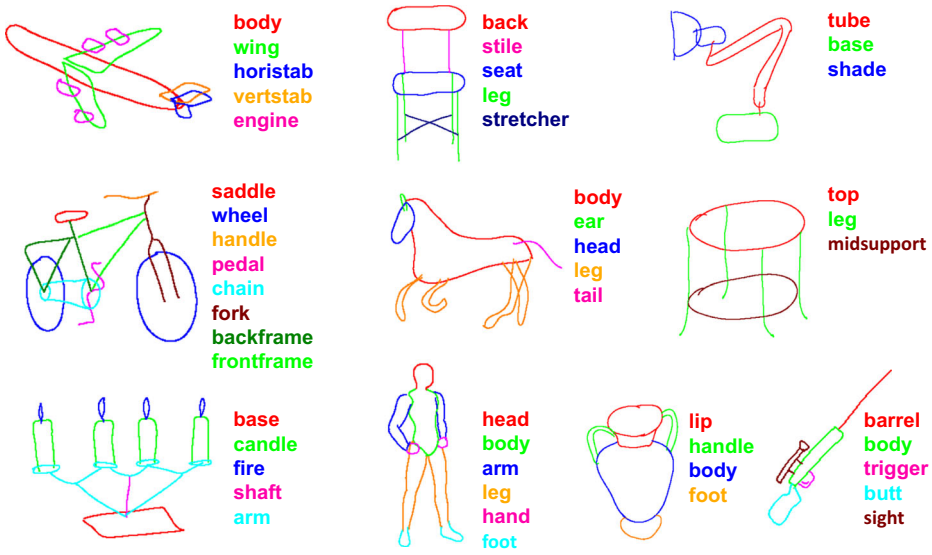


Fig. 5 Our results on Huang dataset

Both we and Li-FCN follow Schneider-CRF's stroke splitting strategy for calculating the component metric accuracies (in Table 2). Specifically, we assign a stroke's all segments with the same label. The label is identical to the stroke's label. Compared with Huang-Auto and Schneider-CRF, Ours-full-3 wins in nine categories and acquires the best average accuracy. For the category *rifle*, our accuracy is close to that of Schneider-CRF, e.g., 64.6% vs. 65.1%. Compared with Li-FCN, Ours-full-5 also wins in nine categories and outperforms 3.2% on average. For the category *chair*, our method may need more training samples as Li-FCN does.

Compared with Ours-full-3, Ours-full-5 improves 2% about pixel metric on average (e.g., 82.8% vs. 80.8% in Table 1), and increases 2.7% about component metric on average (e.g., 83.9% vs. 81.2% in Table 2). This improvement demonstrates that having more training samples is helpful for our method. Figure 5 shows representative segmented sketches by Ours-full-5.

Time We discuss running time performance as follows. We use Tensorflow framework with Python API to implement our CNN model, while we implement the CRF refinement using the unoptimized MATLAB toolbox, UGM [22]. To make a fair comparison, we run our full method on CPU only without GPU acceleration during testing. For a sketch, our running time to test it depends on the number of its strokes. On average, our method spends 0.01s to predict a stroke's label. A sketch in the Huang dataset has 2∼30 strokes. Huang et al. [8] reported that their method needs 40 minutes to segment a single sketch, which is implemented in MATLAB with the optimized library Gurobi [46]. Their experimental platform is a PC with an Intel I7 3GHz CPU and 18GB RAM. Schneider et al. [31] reported that their method spends 2-3 minutes for interpreting a sketch, which is implemented in MATLAB with VLFeat and UGM libraries. However, they stated no details about their experimental platform. Li et al. [16] stated that their method needs $\approx$ 0.35s on average to process a sketch. Their Python implementation was tested on a PC with an Intel Core I5 3.2GHz CPU and 8 GB RAM. Our running time performance is of the same order of magnitude as Li et al. [16]'s.

4.2 Evaluation on Schneider dataset

We perform 4-fold cross validation experiments on Schneider dataset, to be consistent with sample splitting as Schneider et al. [31]. The sketches are drawn freely by the crowd. Due to these abstract sketches having no corresponding 3D meshes, Huang et al. [8] and Li et al. [16]'s methods are not available. Schneider dataset contains clean and unclean data. The unclean data means that some strokes belong to unseen labels. Note that our training procedures are performed on clean data.

Firstly, we perform the experiment on cleaned data for testing. All strokes belong to known labels. It is an ideal testing environment. Table 3 shows the comparisons, and best results are in bold. Our full method achieves good results for those abstract sketches (accuracy over 90%), and our approach outperforms Schneider-CRF by $\approx 8\%$ on average. Each technical element of our method takes its effect. Figure 6 a shows some of our results.

Secondly, we also want to evaluate our method in a real-world environment. Therefore, we use the uncleaned data for testing. Some strokes belong to unseen labels. Though those strokes are always labeled with wrong labels, we still produce good results for strokes with known labels. Specifically, Figure 6b shows some segmented unclean sketches. The strokes with unknown labels (*turbine*, *hair*, and *besom*) are misclassified, but the rest strokes are assigned with suitable labels. The overall accuracy is satisfactory (Table 3), which is close to the accuracy on cleaned data. The gap of average accuracy between clean and unclean data is only 0.9% (e.g., 91.9% vs. 91.0%), while that of Schneider-CRF is 3.6% (e.g., 84.1% vs. 80.5%).

4.3 Network study

We conduct 3-fold cross validation experiments on Huang dataset. Firstly, we perform ablation studies on our network. The inputs are the integration of stroke and sketch.

Table 3 Accuracy results on Schneider dataset

Metric	Method	Airplane	Butterfly	Face	Flower	Pineapple	Snowman	Average
	Schneider-CRF [31]	77.6	78.8	88.3	78.3	96.2	85.3	84.1
Pixel acc. (%)	Ours-C	**90.2**	89.0	92.5	**85.7**	98.3	86.1	90.3
tested on clean data	Ours-CS	88.2	**94.0**	94.1	84.6	98.9	**90.9**	91.8
	Ours	89.7	93.2	**95.1**	84.0	**98.9**	90.3	**91.9**
	Schneider-CRF [31]	78.3	78.6	84.2	73.7	96.0	81.8	82.1
Component acc. (%)	Ours-C	89.6	87.7	91.5	83.5	97.7	81.2	88.5
tested on clean data	Ours-CS	**89.6**	92.5	**96.0**	81.4	98.4	86.7	90.8
	Ours	89.4	**93.1**	94.4	**83.5**	**98.6**	**88.4**	**91.2**
	Schneider-CRF [31]	74.5	78.8	75.6	77.8	96.2	80.0	80.5
Pixel acc. (%)	Ours-C	**90.2**	87.7	92.3	**84.8**	98.3	86.1	89.9
tested on unclean data	Ours-CS	87.9	92.4	93.7	82.5	98.9	**90.3**	90.9
	Ours	87.9	**92.8**	**94.0**	82.4	**98.9**	90.1	**91.0**

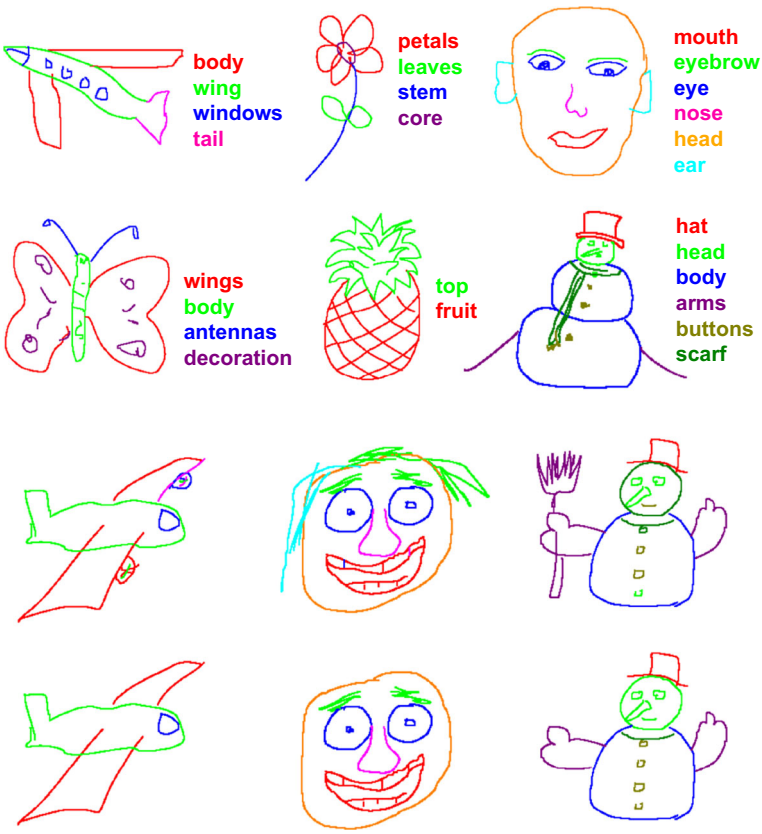


Fig. 6 Our results on Schneider dataset

(i) **First layer filters.** We replace our *conv*15 by the other four filters: *conv*3, *conv*7, *conv*11, and *conv*19. The *conv*{n} means that the sizes of the first layer filters are n × n. The *conv*3, *conv*7, and *conv*11 are adopted in VGGNet [34], ResNet [5], and AlexNet [10] respectively. The *conv*19 is a larger one than the *conv*15. Note Table 4, the *conv*15 wins in six categories on both pixel accuracy and component accuracy, and acquires the best average accuracy. The *conv*3 wins in two categories on pixel accuracy and one category on component accuracy, and acquires the second best average accuracy.

(ii) **Stride for the first layer filters.** We replace our *stride*4 by the other three strides: *stride*3, *stride*5, *stride*6. The *stride*{n} means that the stride of the first layer filters is n. The *stride*3 is used in Sketch-a-Net [47]. The *stride5* and *stride*6 are larger than *stride*4. Note Table 5, the *stride*4 wins in seven categories on pixel accuracy and eight categories on component accuracy, and acquires the best average accuracy. The *stride*6 wins in the rest categories, and acquires the second best average accuracy.

(iii) **Pooling size.** We replace our *pool*3 by the other three pooling layers: *pool*2, *pool*4, and *pool*5. The *pool*{n} means n × n max pooling. The *pool*2 is used in both VGGNet and ResNet. The *pool*4 and *pool*5 are larger than *pool*3. Note Table 6, the *pool*3 wins in five categories on pixel accuracy and seven categories on component accuracy, and

Table 4 Ablation study on first layer filters

Category	Pixel accuracy					Component accuracy				
	conv3	conv7	conv11	conv15	conv19	conv3	conv7	conv11	conv15	conv19
Airplane	80.6	79.7	78.7	**81.1**	78.5	75.6	77.0	73.3	**77.1**	72.1
Bicycle	**88.2**	84.7	87.1	84.0	83.0	**81.9**	79.1	81.0	78.7	77.1
Cdlbrm	70.4	66.8	68.5	**72.2**	66.9	66.0	62.9	64.4	**69.6**	63.8
Chair	66.6	68.0	64.1	**70.7**	64.9	60.5	61.6	54.5	**63.8**	58.8
Fourleg	84.7	**86.0**	83.8	84.6	82.6	79.1	**81.1**	78.2	79.3	76.0
Human	**79.1**	78.6	77.0	78.6	73.8	72.3	**72.6**	70.9	72.1	68.0
Lamp	93.6	94.2	92.5	**95.0**	94.8	93.9	92.9	92.7	**95.1**	95.1
Rifle	71.2	67.3	**71.3**	68.3	68.1	66.4	62.6	**67.4**	64.5	62.7
Table	79.1	76.9	81.0	**82.3**	79.8	72.4	71.4	73.1	**76.7**	72.0
Vase	85.0	85.8	84.5	**86.9**	84.5	80.5	80.5	80.2	**84.2**	81.0
Average	79.8	78.8	78.8	**80.4**	77.7	74.9	74.2	73.6	**76.1**	72.7

The best results are in bold

acquires the best average accuracy. The *pool*5 wins in three categories on both pixel accuracy and component accuracy, and acquires the second best average accuracy.

Secondly, we compare our network with four mainstream networks: AlexNet [10], Sketch-a-Net [47], VGGNet [34], and ResNet [5]. Sketch-a-Net is pre-trained on TU-Berlin dataset [2] for sketch recognition, and the other three networks are pre-trained on ImageNet

Table 5 Ablation study on stride for the first layer filters

Category	Pixel accuracy				Component accuracy			
	stride3	stride4	stride5	stride6	stride3	stride4	stride5	stride6
Airplane	77.6	**81.1**	77.8	78.8	72.4	**77.1**	72.6	73.6
Bicycle	82.8	84.0	86.1	**86.9**	77.4	78.7	80.4	**81.5**
Cdlbrm	66.5	**72.2**	65.6	69.6	63.1	**69.6**	62.9	65.0
Chair	65.1	**70.7**	67.8	67.0	55.9	**63.8**	59.7	59.7
Fourleg	81.4	**84.6**	83.7	81.8	76.9	**79.3**	77.5	77.0
Human	73.8	78.6	78.2	**78.9**	68.1	72.1	72.0	**72.7**
Lamp	90.5	**95.0**	94.5	94.4	88.7	**95.1**	93.5	93.4
Rifle	68.1	68.3	68.0	**70.0**	64.5	64.5	64.5	**65.5**
Table	78.0	**82.3**	80.7	81.0	72.3	**76.7**	74.1	74.7
Vase	83.5	**86.9**	83.9	83.8	80.2	**84.2**	82.3	81.8
Average	76.7	**80.4**	78.6	79.2	71.9	**76.1**	73.9	74.5

The best results are in bold

Table 6 Ablation study on pooling size

Category	Pixel accuracy				Component accuracy			
	pool2	pool3	pool4	pool5	pool2	pool3	pool4	pool5
Airplane	78.6	**81.1**	76.6	72.2	74.2	**77.1**	72.4	77.0
Bicycle	82.2	84.0	**85.4**	80.3	75.5	78.7	79.5	**80.5**
Cdlbrm	64.3	**72.2**	69.4	74.8	60.6	**69.6**	65.1	67.3
Chair	64.5	**70.7**	67.7	66.4	57.0	**63.8**	59.1	61.2
Fourleg	80.3	84.6	85.6	**86.6**	74.8	79.3	79.3	**80.9**
Human	74.4	78.6	**78.7**	83.0	67.6	72.1	73.4	**73.9**
Lamp	91.0	**95.0**	94.7	93.7	91.1	**95.1**	93.8	91.7
Rifle	67.3	68.3	67.2	**74.1**	62.8	**64.5**	64.3	63.2
Table	75.9	**82.3**	80.5	79.0	69.6	**76.7**	74.4	75.8
Vase	84.0	86.9	85.4	**87.3**	79.7	**84.2**	82.9	82.6
Average	76.2	**80.4**	79.1	79.7	71.3	**76.1**	74.4	75.4

The best results are in bold

dataset [26] for natural image recognition. We fine-tunes these networks on Huang dataset for stroke recognition. Table 7 compares their results, parameters, floating point operations (FLOPs), and time costs. Our average result surpasses the other results. The ResNet-50 have equivalent performance to ours, but our network only has 0.6 million parameters, fewer than that of the ResNet-50. To predict a stroke's label, our network spends 8 ms on CPU, while the ResNet-50 needs 136 ms.

5 Limitation and future work

Our work assumes that each stroke is the basic component for assigning a label. However, when people draw a stroke, the stroke may cover multiple components. Huang et al. [8] and Schneider et al. [31]'s stroke splitting strategies may help to solve this, but their strategies result in a large number of unnecessary segments. The ideal solution is only splitting those strokes that cover multiple components. How to identify the strokes is a research direction.

Our running time to process a sketch depends on the number of its strokes. On average, our method spends 10 ms to process a stroke. In practical sketch-based interfaces, a sketch includes one hundred strokes at most. Our method spends around 1 s to process the sketch. The computing time is also acceptable. To improve time performance, predicting all the strokes at once may be explored.

The training procedures of CNN and CRF are separated, and the edge features in the CRF model are simple. This provides a strong motivation for exploring powerful edge descriptive feature learning, and end to end frameworks based on graph convolutional networks [4, 33, 37].

Table 7 Comparison with four mainstream CNNs

CNN	Params	FLOPs	CPU	Metric	Airplane	Bicycle	Cdlbrm	Chair	Fourleg	Human	Lamp	Rifle	Table	Vase	Average
AlexNet	62.3 M	727 M	39 ms		78.8	84.6	68.7	63.7	84.9	77.5	89.5	68.1	77.1	84.8	77.8
Sketch-a-Net	8.6 M	681 M	24 ms	Pixel	78.3	85.1	68.9	61.2	82.5	76.6	91.0	64.7	77.1	83.5	76.9
VGGNet-16	138.3 M	16 G	235 ms	acc.	80.3	85.9	72.3	68.4	80.6	78.4	93.2	70.9	77.4	86.7	79.4
ResNet-50	25.6 M	4 G	136 ms		81.6	86.1	71.0	70.0	85.6	80.0	95.2	67.8	76.0	89.1	80.2
Ours-CS	**0.6 M**	**162 M**	**8 ms**		81.1	84.0	72.2	70.7	84.6	78.6	95.0	68.3	82.3	86.9	80.4
AlexNet	62.3 M	727 M	39 ms		73.3	77.0	64.2	57.0	79.2	71.8	88.5	64.4	71.9	80.5	72.8
Sketch-a-Net	8.6 M	681 M	24 ms	Comp.	72.2	77.4	64.3	53.9	77.0	71.1	88.9	61.5	70.9	81.2	71.8
VGGNet-16	138.3 M	16 G	235 ms	acc.	74.9	78.7	69.8	61.7	75.8	73.0	91.0	64.7	69.0	84.1	74.3
ResNet-50	25.6 M	4 G	136 ms		77.4	79.3	65.5	64.2	81.2	74.4	93.7	64.6	66.4	84.6	75.1
Ours-CS	**0.6 M**	**162 M**	**8 ms**		77.1	78.7	69.6	63.8	79.3	72.1	95.1	64.5	76.7	84.2	76.1

Their inputs are the integration of stroke and sketch. The best results are in red, and the second best results are in blue

The best results are in bold

6 Conclusion

We have proposed a novel hybrid approach based on CNN and CRF for automatic 2D sketch labeling. The method directly works on sketch domain and needs no additional labeled 3D meshes. It is effective for both abstract and 3D-styled sketches. The experimental results demonstrate that our method achieves the state-of-the-art level on both accuracy and runtime.

Acknowledgements The work is supported by the National Key R&D Program of China (2018YFB0203904), NSFC from PRC (61872137, 61502158, 61803150), Hunan NSF (2017JJ3042, 2018JJ3067).

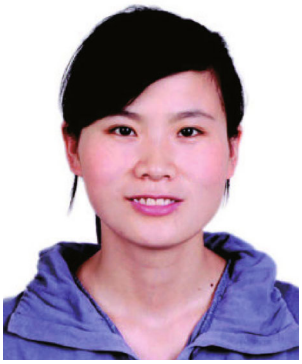
References

1. Besag J (1986) On the statistical analysis of dirty pictures. J R Stat Soc Ser B Methodol 48(3):259–302
2. Eitz M, Hays J, Alexa M (2012) How do humans sketch objects? ACM Trans. Graph 31(4):44:1–44:10
3. Fan L, Wang R, Xu L, Deng J, Liu L (2013) Modeling by drawing with shadow guidance. Comput Graphics Forum 32(7):157–166
4. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J, Chen T (2018) Recent advances in convolutional neural networks. Pattern Recogn 77:354–377. https://doi.org/10.1016/j.patcog.2017.10.013
5. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE conference on computer vision and pattern recognition, pp 770–778
6. He JY, Wu X, Jiang YG, Zhao B, Peng Q (2017) Sketch recognition with deep visual-sequential fusion model. In: Proceedings of the 2017 ACM on multimedia conference. ACM, pp 448-456
7. Hu M, Ou B, Xiao Y (2017) Efficient image colorization based on seed pixel selection. Multimedia Tools Appl 76(22):23567–23588
8. Huang Z, Fu H, Lau RW (2014) Data-driven segmentation and labeling of freehand sketches. ACM Trans Graph 33(6):175:1–175:10
9. Kim B, Wang O, Öztireli AC, Gross M (2018) Semantic segmentation for line drawing vectorization using neural networks. Comput Graphics Forum 37(2):329–338
10. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
11. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data
12. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
13. Li SZ (1994) Markov random field models in computer vision. In: European conference on computer vision. Springer, pp 361–370
14. Li B, Lu Y, Johan H, Fares R (2017) Sketch-based 3d model retrieval utilizing adaptive view clustering and semantic information. Multimed Tools Appl 76(24):26603–26631
15. Li Y, Lei H, Lin S, Luo G (2018) A new sketch-based 3d model retrieval method by using composite features. Multimed Tools Appl 77(2):2921–2944
16. Li L, Fu H, Tai C (2019) Fast sketch segmentation and labeling with deep learning. IEEE Comput Graph Appl 39(2):38–51. https://doi.org/10.1109/MCG.2018.2884192
17. Liu L, Wiliem A, Chen S, Lovell BC (2016) Automatic and quantitative evaluation of attribute discovery methods. In: 2016 IEEE winter conference on applications of computer vision, WACV 2016, Lake Placid, NY, USA, March 7-10, pp 1–9
18. Liu L, Shen F, Shen Y, Liu X, Shao L (2017) Deep sketch hashing: fast free-hand sketch-based image retrieval. In: Proceedings of CVPR, pp 2862–2871
19. Liu L, Wiliem A, Chen S, Lovell BC (2017) What is the best way for extracting meaningful attributes from pictures? Pattern Recogn 64:314–326
20. Liu L, Nie F, Wiliem A, Li Z, Zhang T, Lovell BC (2018) Multi-modal joint clustering with application for unsupervised attribute discovery. IEEE Trans Image Process 27(9):4345–4356
21. Lowe DG (1999) Object recognition from local scale-invariant features. In: IEEE international conference on computer vision. IEEE, pp 1150–1157

22. Mark S (2015) UGM: Matlab code for undirected graphical models. http://www.cs.ubc.ca/schmidtm/Software/UGM.html
23. Noris G, Sỳkora D, Shamir A, Coros S, Whited B, Simmons M, Hornung A, Gross M, Sumner R (2012) Smart scribbles for sketch segmentation. Comput Graphics Forum 31(8):2516–2527
24. Qi Y, Guo J, Li Y, Zhang H, Xiang T, Song YZ (2013) Sketching by perceptual grouping. In: 2013 20th IEEE international conference on image processing (ICIP). IEEE, pp 270–274
25. Qi Y, Song YZ, Xiang T, Zhang H, Hospedales T, Li Y, Guo J (2015) Making better use of edges via perceptual grouping. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1856–1865
26. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein MS, Berg AC, Li F (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
27. Sánchez J, Perronnin F, Mensink T, Verbeek J (2013) Image classification with the fisher vector: theory and practice. Int J Comput Vis 105(3):222–245
28. Sangkloy P, Burnell N, Ham C, Hays J (2016) The sketchy database: learning to retrieve badly drawn bunnies. ACM Trans Graph (TOG) 35(4):119
29. Sangkloy P, Lu J, Fang C, Yu F, Hays J (2017) Scribbler: controlling deep image synthesis with sketch and color. In: IEEE conference on computer vision and pattern recognition (CVPR), vol 2
30. Sarvadevabhatla RK, Dwivedi I, Biswas A, Manocha S et al (2017) Sketchparse: towards rich descriptions for poorly drawn sketches using multi-task hierarchical deep networks. In: Proceedings of the 2017 ACM on multimedia conference. ACM, pp 10–18
31. Schneider RG, Tuytelaars T (2016) Example-based sketch segmentation and labeling using crfs. ACM Trans Graph 35(5):151:1–151:9
32. Seddati O, Dupont S, Mahmoudi S (2017) Deepsketch 3. Multimed Tools Appl 76(21):22333–22359
33. Shang C, Liu Q, Chen KS, Sun J, Lu J, Yi J, Bi J (2018) Edge attention-based multi-relational graph convolutional networks. arXiv preprint arXiv:180204944
34. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations
35. Sun Z, Wang C, Zhang L, Zhang L (2012) Free hand-drawn sketch segmentation. In: European conference on computer vision. Springer, pp 626–639
36. Tan G, Chen H, Qi J (2016) A novel image matting method using sparse manual clicks. Multimed Tools Appl 75(17):10213–10225
37. Tompson JJ, Jain A, LeCun Y, Bregler C (2014) Joint training of a convolutional network and a graphical model for human pose estimation. In: Advances in neural information processing systems, pp 1799–1807
38. Wan L, Xiao Y, Dou N, Leung C, Lai Y (2018) Scribble-based gradient mesh recoloring. Multimed Tools Appl 77(11):13753–13771
39. Wang C, Yang H, Bartz C, Meinel C (2016) Image captioning with deep bidirectional lstms. In: Proceedings of the 24th ACM international conference on multimedia. ACM, pp 988–997
40. Wang C, Yang H, Meinel C (2016) A deep semantic framework for multimodal representation learning. Multimed Tools Appl 75(15):9255–9276
41. Wang C, Niepert M, Li H (2018) LRMM: learning to recommend with missing modalities. In: Proceedings of the 2018 conference on empirical methods in natural language processing, Brussels, Belgium, October 31 - November 4, pp 3360–3370
42. Wang C, Yang H, Meinel C (2018) Image captioning with deep bidirectional lstms and multi-task learning. ACM Trans Multimed Comput Commun Appl (TOMM) 14(2s):40
43. Wang SH, Muhammad K, Hong J, Sangaiah AK, Zhang YD (2019) Alcoholism identification via convolutional neural network based on parametric relu, dropout, and batch normalization. Neural Comput & Applic, pp 1–16. https://doi.org/10.1007/s00521-018-3924-0
44. Xu K, Chen K, Fu H, Sun WL, Hu SM (2013) Sketch2scene: sketch-based co-retrieval and co-placement of 3d models. ACM Trans Graph (TOG) 32(4):123
45. Xu B, Chang W, Sheffer A, Bousseau A, McCrae J, Singh K (2014) True2form: 3d curve networks from 2d sketches via selective regularization. ACM Trans Graph 33(4):131:1–131:13
46. Yin W (2009) Gurobi mex: a matlab interface for gurobi. http://convexoptimization.com/wikimization/index.php/gurobi_mex
47. Yu Q, Yang Y, Liu F, Song YZ, Xiang T, Hospedales TM (2017) Sketch-a-net: a deep neural network that beats humans. Int J Comput Vis 122(3):411–425
48. Zhang YD, Muhammad K, Tang C (2018) Twelve-layer deep convolutional neural network with stochastic pooling for tea category classification on gpu platform. Multimed Tools Appl 77(17):22821–22839
49. Zheng Y, Cao X, Xiao Y, Zhu X, Yuan J (2019) Joint residual pyramid for joint image super-resolution. J Vis Commun Image Represent 58:53–62
50. Zhou S, Zhou C, Xiao Y, Tan G (2018) Patchswapper: a novel real-time single-image editing technique by region-swapping. Comput Graph 73:80–87

Xianyi Zhu is currently a PhD. student at the College of Computer Science and Electronic Engineering, Hunan University, China. He received his Master's degree in computer science from Hunan University in 2016. His research interests include computer vision and computer graphics.

Yi Xiao received the Bachelor's degree and Master's degree in Mathematics from Sichuan University in 2005 and 2008, and the PhD. degree in Electronic Engineering from City University of Hong Kong in 2012. He is currently an Associate Professor in college of Computer Science and Electronic Engineering, Hunan University, China. His research interests include computer graphics, GPU computing and neural networks.



Yan Zheng received the Bachelor's degree from Department of Mathematics, Hebei Normal University in 2006, Master's degree from Department of Mathematics, Sichuan University in 2009, and the PhD. degree degree in Department of Mechanical and Biomedical Engineering, City University of Hong Kong in 2012. She is currently an Assistant Professor in the college of Electrical and Information Engineering, Hunan University, China. Her research interests include switched system, positive systems and image.