SketchLattice: Latticed Representation for Sketch Manipulation

Yonggang Qi\textsuperscript{1*}  Guoyao Su\textsuperscript{1*}  Pinaki Nath Chowdhury\textsuperscript{2}  Mingkang Li\textsuperscript{1}  Yi-Zhe Song\textsuperscript{2}

\textsuperscript{1}Beijing University of Posts and Telecommunications, CN  \textsuperscript{2}SketchX, CVSSP, University of Surrey, UK

\{qiyg, sgybupt, lmk\}@bupt.edu.cn \{p.chowdhury, y.song\}@surrey.ac.uk

Abstract

The key challenge in designing a sketch representation lies with handling the abstract and iconic nature of sketches. Existing work predominantly utilizes either, (i) a pixelative format that treats sketches as natural images employing off-the-shelf CNN-based networks, or (ii) an elaborately designed vector format that leverages the structural information of drawing orders using sequential RNN-based methods. While the pixelative format lacks intuitive exploitation of structural cues, sketches in vector format are absent in most cases limiting their practical usage. Hence, in this paper, we propose a lattice structured sketch representation that not only removes the bottleneck of requiring vector data but also preserves the structural cues that vector data provides. Essentially, sketch lattice is a set of points sampled from the pixelative format of the sketch using a lattice graph. We show that our lattice structure is particularly amenable to structural changes that largely benefits sketch abstraction modeling for generation tasks. Our lattice representation could be effectively encoded using a graph model, that uses significantly fewer model parameters (13.5 times lesser) than existing state-of-the-art. Extensive experiments demonstrate the effectiveness of sketch lattice for sketch manipulation, including sketch healing and image-to-sketch synthesis.

1. Introduction

Research on freehand human sketches has become increasingly popular in recent years. Due to its ubiquitous ability in recording visual objects [6], sketches form a natural medium for human-computer interaction. Deriving a tailor-made representation for sketches sits at the core of sketch research, and has direct impact on a series of downstream applications such as sketch recognition [33, 27, 12], sketch-based image retrieval [3, 32, 17, 16], sketch-3D reconstruction [15, 9, 25, 19, 22], and sketch synthesis [14, 23]. Albeit a pivotal component, designing an effective representation is challenging since sketches are typically abstract and iconic.

Prior works predominantly relied on encoding sketch in a pixelative format (i.e., an image) [6, 33, 21]. Although it provided the convenience of using off-the-shelf convolutional neural network effortlessly re-purposed for sketches, pixelative format lacks the intuitive exploitation of structural information. The presence of structural information is vital for sketch abstraction modeling [6, 20], which in turn is essential for downstream tasks that dictate structural manipulation such as sketch generation [7, 4, 10] and sketch synthesis [14, 23].

RNN-based approaches have consequently emerged as means to fully explore the sequential nature of sketches [7]. The research convention is to use \texttt{QuickDraw} [8] vector format where each sketch is represented as a list of offsets in $x$ and $y$. Thanks to the stroke-level modeling, these approaches do offer a degree of flexibility in generation and synthesis tasks, yet they do so by imposing a strong assumption – all sketches have sequential stroke data to them. This assumption largely prohibits the application of RNN-based methods to work with sketches such as those drawn on a piece of paper. A natural question is therefore – is there a way to remove the bottleneck on requiring vector data but

\textsuperscript{*}Equal contribution
There are mainly two types of data formats for sketch representation—image-based and sequential representation. While the former treats sketch as a conventional 2D image with pixel values (i.e., the pixelative format), the latter considers sketch as an elaborately designed set of ordered stroke points (i.e., vector format), represented by offset coordinates along x and y directions with pen states (touch, lift and end) [7]. Traditional sketch feature extractors are usually CNN-based approaches [33, 4] that can directly take sketch image in pixelative format during input. However, this is highly redundant due to the sparsity of line drawings in a sketch image that necessitates heavy engineering efforts [33]. Additionally, CNN-based approaches cannot effectively capture structural cues since it does not encode position and orientation of objects, leading to sub-par results on generation models.

In contrast, sequential representation is sketch-specific [8], designed according to the drawing habit of humans which is constructed stroke-by-stroke. Such sequential, vector format representation allows modeling sketches using RNN-based methods. This resulted in impressive results such as sketch generation and sketch synthesis using long short-term memory (LSTM) [7, 24]. Although promising, such RNN-based approaches require a vectorized data format at the input, which leads to a major bottleneck limiting practical usage in the absence of vector sketches, like sketches drawn on a piece of paper. Therefore, we aim to propose an unexplored technique, by employing a more practical lattice sketch representation, i.e., SketchLattice, that avoids storing stroke orders while still keeping strong spatial evidence that vector sketches typically provide.

Graphical Sketch Embedding Graph convolutional networks (GCNs) [2, 12] were originally designed to deal with structured data, such as knowledge graphs or social networks, by generalizing neural networks on graphs. In the past few years, exciting developments have been made that explore GCNs capabilities for various vision tasks including image classification [5], captioning [31], image understanding [1], action recognition [13], 3D object detection [34], and shape analysis [26]. Yet, until recently, a few attempts [30, 29] started to apply GCNs on sketch embedding. The existing visual sparsity and spatial structure of sketch strokes are naturally compatible with graphical representations. However, the dominating approach in sketch research assumes access to a vector format where the stroke orders are required, thus resulting in a major limitation in real-world cases. On contrary, ours provides a generic approach that explores the geometrical proximity in a lattice representation of sketch. We also show how our proposed graphical sketch embedding can be used additionally for tasks, involving sketch generation and image-to-skeleton translation.

2. Related Work

Sketch Data Format There are mainly two types of data formats for sketch representation—image-based and sequential representation. While the former treats sketch as...
nique that (i) transforms an input 2D sketch image \( S \) into a set of points \( S^L = \{p_1, p_2, \ldots, p_m\} \), using a lattice graph \( F_{\text{lattice}} \). Each point \( p_i = (x, y) \) in \( S^L \) represents the absolute coordinates \( x \) and \( y \) in the \( S \). We call \( S^L \) as the lattice format representation of \( S \). (ii) Our novel lattice format \( S^L \) could be seamlessly transformed to a graphical form \( G = (V, E) \) that is encoded into a \( d \)-dimensional sketch-level embedding vector \( \Psi \in \mathbb{R}^d \) using a simple off-the-shelf GCN-based model. (iii) We observe how this sketch-level embedding vector \( \Psi \) could help in downstream tasks such as sketch generation by using existing LSTM-based decoding models. Figure 2 offers a schematic illustration.

### 3.1. Latticed Sketch

The input to our proposed latticed sketch representation is a sketch image \( S \in \mathbb{R}^{w \times h} \) where \( w \) and \( h \) represent the width and height of \( S \) respectively. We extract the latticed sketch \( S^L \) from \( S \) using the lattice graph \( F_{\text{lattice}} \). Our lattice graph \( F_{\text{lattice}} \) is a grid that constitutes of uniformly distributed \( 2n \) horizontal and vertical lines, arranged in a criss-cross manner. The optimal value of \( n \) for any given sketch image \( S \), could be empirically determined during inference without further training. As shown in Figure 2, we construct \( S^L \) by sampling the set of all overlapping points \( S^L = \{p_1, p_2, \ldots, p_m\} \) between a black pixel in sketch image \( S \) representing a stroke region, and the \( 2n \) horizontal or vertical lines in \( F_{\text{lattice}} \). Formally, we define \( S^L \) as:

\[
S^L = F_{\text{lattice}}(S)
\]

Although extremely simple, this novel latticed sketch representation \( S^L \) is very informative since it can express the topology (i.e., overall structure and shape) of the original sketch image \( S \), without the need of vector data. Additionally, our latticed sketch representation is very flexible because, (i) there is no constrain for the size of input sketch image \((w \times h)\) and the original aspect ratio is maintained, (ii) the abstraction level of the generated sketches modulates depending on the sampling density from the lattice graph \( F_{\text{lattice}} \) by varying the value of \( n \). Increasing the value of \( n \) would result in more detailed sketches, whereas decreasing it would lead to highly abstract sketches. Figure 1(c) and 4 demonstrate how adding more sample points \( p_i \) changes the abstraction level of generated sketches.

### 3.2. Graph Construction

**Graph Nodes** \( V \) The SketchLattice \( S^L \) can be effectively encoded by a simple graph model which not only consumes fewer model parameters, thereby increasing efficiency, but also allows easier optimization resulting in a model better trained to give a state-of-the-art performance. For each point \( p_i \in S^L \) we calculate \( v_i \in V \) representing elements of the set \( V \), denoting graph nodes. To ensure that the encoding process is amenable to structural changes, each point \( p_i \) is tokenized by a learnable embedding function \( F_{\text{emb}}(\cdot) : \mathbb{R}^2 \rightarrow \mathbb{R}^d \) that maps the absolute point location \( p_i = (x, y) \) to a \( d \)-dimensional vector space. Formally,

\[
v_i = F_{\text{emb}}(p_i)
\]

where \( v_i \in \mathbb{R}^d \) is the resulting tokenized vector representation. Maintaining the original aspect ratio, we resize and pad the input sketch image \( S \in \mathbb{R}^{w \times h} \) to a size of \((256, 256)\) before applying the lattice graph \( F_{\text{lattice}} \). Hence, the vocabulary size of the learned embedding function \( F_{\text{emb}} \) is 256\(^2\).

Our intuition is that, by using an embedding function \( F_{\text{emb}} \) that tokenizes each point location \( (x, y) \) to a \( d \)-dimensional vector \( v_i \), the model would learn to obtain similar embedding features for nearby points. Hence the resulting representation would be more robust to the frequently observed shape deformation and be more amenable to largely benefits sketch abstraction modeling in the generation tasks.

**Graph Edges** \( E \) A straight-forward yet efficacious approach based on the geometric proximity principles is adopted to construct the graph edge links among nodes, based on the corresponding latticed points’ locations \( p_i =
is projected into two vectors $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}^d$, then from which we can sample a random vector $z \in \mathbb{R}^d$ by using the reparameterization trick [11] to introduce stochasticity in the generation process via an IID Gaussian variable $\mathcal{N}(0, I)$:

$$ z = \mu + \sigma \odot \mathcal{N}(0, I) $$

where $W_\mu$ and $W_\sigma$ are learned through backpropagation [7]. The latent vector $z$ is used as a condition for the LSTM decoder to sequentially predict sketch strokes. Specifically, the output stroke representation $s_{t-1}$ from the previous time step, together with latent vector $z$ serve as inputs to update the LSTM hidden state $h_{t-1}$ by:

$$ h_t = LSTM_{forward}(h_{t-1}; [s_{t-1}, z]) $$

where $[\cdot]$ represents concatenation operation. Next, a linear layer is used to predict a output stroke representation for current time step, i.e., $s_t = W_s h_t + b_s$, where $W_s$ and $b_s$ are learnable weight and bias. The final stroke coordinates are derived from $s_t$ with the help of Gaussian mixture models, to generate the vector sketch format, represented by $S'$. We refer readers to [7, 8] for more details.

### 3.5. Model Training and Deployment

Our proposed graphical sketch encoder and the generative LSTM decoder are trained end-to-end for sketch generation. Note that, although we require vector sketches to train the LSTM decoder for the purpose of vector sketch generation, our model fully works on image sketch input, rather than vector data during inference. Following [7], the goal is to minimize the negative log-likelihood of the generated probability distribution to explain the training data $S$, which can be defined as:

$$ \min_{\theta(z|s)} [-\log p_\theta(S|z)] $$

which seeks to reconstruct the vector sketch representation $S$ from the predicted latent vector $z$. Upon training, the decoder generates a vector sketch conditioned on the graphical encoded latent vector $z$, obtained from our lattice sketch $S^L$ given any image sketch, thus being more effective for practical applications.

### 4. Experiments

The ability to be amenable to structural changes and enable appropriate abstraction modeling for sketch generation are the two key aspects that our proposed SketchLattice representation aims to address. Specifically, we adopt the challenging task of sketch healing to testify the robustness of our novel sketch representation towards frequently occurring structural deformation in sketches. Additionally, we
also observe how our proposed approach could be utilized to perform the task of image-to-sketch translation.

**Implementation details** We implement our model on PyTorch [18] using a single Nvidia Tesla T4 GPU. Optimization is performed using the Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$. Value of learning rate is set to $10^{-3}$ along with a decay rate of 0.999 in every iteration. A gradient clipping strategy is adopted to prevent gradient from exploding during the training of LSTM decoder. Essentially, we force the gradient value to 1.0 if the actual value is larger than 1.0. The optimal value for the number of graph encoding layer is $K = 2$.

**4.1. Sketch Healing**

The task of sketch healing [24] was proposed akin to vector sketch synthesis. Specifically, given a partial sketch drawing, the objective is to recreate a sketch which can best resemble the partial sketch.

**From full $S^L$ to partial $\hat{S}^L$** Given the lattice sketch representation $S^L$ from an input sketch image $S$, we randomly drop a fraction of lattice points in $S^L$ with some probability $P_{\text{mask}}$ to generate a partial SketchLattice, represented by $\hat{S}^L$. Hence, the graph edges linked to the removed node are also disconnected, thereby simultaneously modifying the adjacency matrix $A$. One can think of $P_{\text{mask}}$ as the corruption level of an input sketch image.

**4.1.1 Experimental Settings**

**Dataset** Following the footsteps of [27, 24], we use QuickDraw [8] for evaluation since it is currently the largest doodle sketch dataset. More specifically, a small subset of 10 categories are selected such that it includes (i) both complex and simple drawings, (ii) intra-category objects having high degree of resemblance among each other, and (iii) presence of common life object categories that contain diverse sub-categories such as bus and umbrella. In each category we use 70k training and 1k testing sketches. The selected 10 categories are as follows: airplane, angel, apple, butterfly, bus, cake, fish, spider, The Great Wall, umbrella.

**Competitors** We compare our proposed Lattice-GCN-LSTM network with three most popular alternatives for vector sketch generation: SketchRNN (SR) [7], SketchPix2seq (Sp2s) [4], and SketchHealer (SH) [24]. Input to SketchRNN is a set of offsets in $x$ and $y$ directions from a vector sketch representation. The key to SketchRNN is a sequence-to-sequence model which is trained without the KL-divergence term. This is done to maintain the fairness of comparison since the KL-divergence term has shown to be beneficial for multi-class scenarios [4]. SketchPix2seq on the other hand replaces its encoding module with a CNN-based encoder that accepts a pixelative format i.e., sketch image. It is expected that such a design will help capture better visual information. Note that, although both SketchRNN and SketchPix2seq were not specifically designed for the task of sketch healing, following [24], we employ these techniques since they are procedurally-wise compatible after being re-purposed. The only work specifically addressing the task of sketch healing is SketchHealer [24].

Given a vector sketch as input, SketchHealer converts it into a graphical form where each stroke is considered as a node. Next, visual image patches are extracted from each node region. A GCN-based model is applied for encoding a random vector $z$. The decoding procedure of SketchHealer is identical to ours where $z$ is fed into a generative LSTM decoder to generate the corresponding vector sketch. Additionally, we retrain a variant of SketchHealer only using visual cues (SH-VC), for which stroke order is unavailable. Hence, geometric proximity principles are utilized to construct graph edges, similar to ours. This examines the performance of SH [24] in the absence of vector sketches.

**Evaluation setup** We adopt a similar evaluation setup in [23] and [24] for quantitative evaluation to understand the effectiveness of our novel latticed sketch representation. First, we evaluate the quality of generated vector sketches (transformed to pixelative format), via sketch recognition accuracy. A pre-trained multi-category classifier with AlexNet architecture is used, which is trained on the training split of 345 QuickDraw categories. Higher recognition accuracy essentially signifies the ability of the network to generate realistic sketches. It also indicates that the network is able to accurately model the underlying data distribution by effectively encoding a sketch into an accurate and informative sketch representation. We use 1000 testing sketches from each of the 10 selected categories for a thorough evaluation. Second, we judge the recognizability of the encoded sketch-level latent vector $\Psi$ by performing a sketch-to-sketch retrieval task. The objective is, given the encoded representation of a sketch $\Psi$, we expect to retrieve sketches of the same category from a gallery of sketches. A higher retrieval accuracy signifies that the network has a strong sketch healing ability, due to its amenable and robust sketch representation.

**4.1.2 Results**

**Qualitative Results** We illustrate some examples produced by our lattice-based sketch generator under different values of $P_{\text{mask}}$ in Figure 3. We can observe that (i) our latticed representation is robust to partial missing parts such that ours can still generate a novel complete sketch even up to $P_{\text{mask}} = 30\%$. (ii) The generated sketch is sensitive to the number of the obtained sampling points, where more points lead to greater details in the generated sketch. Tak-
Figure 3. Exemplary results of generated sketch from SketchLattice under different corruption level of mask probability $P_{mask}$ in Quick-Draw dataset. With an increase of $P_{mask}$, the generated sketch becomes more abstract. For $P_{mask} \leq 30\%$ we observe satisfactory generated sketches, but for $P_{mask} = 50\%$, the generated new sketches are struggle to faithfully recover the original sketch.

Figure 4. Examples showing how adding more lattice points in different stages (color coded), result in the Lattice-GCN-LSTM network to progressively generate more detailed representation of the category.

Table 1. Recognition accuracy (Acc) and Retrieval results (Top-1) according to different corruption levels ($P_{mask}$). We use “nearby” proximity principle and optimal value of $n = 32$. Notice that neither vector format (VF) input nor visual cues (VC) are required by using our method during test.

<table>
<thead>
<tr>
<th>Method</th>
<th>VF</th>
<th>VC</th>
<th>#Params</th>
<th>$P_{mask}$</th>
<th>Acc</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR [7]</td>
<td>✓</td>
<td>✗</td>
<td>0.67 M</td>
<td>10%</td>
<td>25.08%</td>
<td>50.65%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30%</td>
<td>3.44%</td>
<td>43.48%</td>
</tr>
<tr>
<td>Sp2s [4]</td>
<td>✗</td>
<td>✓</td>
<td>1.36 M</td>
<td>10%</td>
<td>24.26%</td>
<td>45.20%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30%</td>
<td>10.54%</td>
<td>27.66%</td>
</tr>
<tr>
<td>SH [24]</td>
<td>✓</td>
<td>✓</td>
<td>1.10 M</td>
<td>10%</td>
<td>50.78%</td>
<td>85.74%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30%</td>
<td>43.26%</td>
<td>85.47%</td>
</tr>
<tr>
<td>SH-VC [24]</td>
<td>✗</td>
<td>✓</td>
<td>1.10 M</td>
<td>10%</td>
<td>58.48%</td>
<td>58.48%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30%</td>
<td>58.48%</td>
<td>58.48%</td>
</tr>
<tr>
<td>Ours</td>
<td>✗</td>
<td>✗</td>
<td>0.08 M</td>
<td>10%</td>
<td>55.50%</td>
<td>76.02%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30%</td>
<td>54.79%</td>
<td>73.71%</td>
</tr>
</tbody>
</table>

Quantitative Results As discussed in Section 4.1.1, we compare the performance of different models under the two metrics (recognition accuracy and Top-1 retrieval) as shown in Table 1. We can observe from Table 1 that our approach outperforms other baseline methods on recognition accuracy, suggesting that the healed sketches obtained from ours are more likely to be recognized as objects in the correct categories. Importantly, we can also observe that, unlike other competitors which are very sensitive to the corruption level, ours can maintain a stable recognition accuracy even when $P_{mask}$ increases up to 30%. For the task of sketch-to-sketch retrieval, we can see that ours achieves the second best, which is inferior to SketchHealer. However, Sketch-Healer depends heavily on stroke order provided by vectorized sketches, evidenced by the dramatic decrease of retrieval performance (Table 1 SH vs SH-VC). This signifies the importance and superiority of our approach in the absence of vector input, which is a common practice in real-world cases. In addition, our network is much lighter than the other competitors having far less parameters (13.5 times lesser than SketchHealer), since our approach avoids the use of expensive CNN-based operation. We further include two classes circle and clock as distraction to apple for evaluation. A slightly better result 77.80% (vs 76.02%) can be observed (12 classes, $P_{mask}=10\%$), suggesting good
Figure 5. Qualitative retrieval results (Top 6) for sketch-to-sketch retrieval for QuickDraw categories under different corruption levels of $P_{\text{mask}}$. Red bounding boxes denote false positive.

Figure 6. T-SNE plot on sketch-level latent vector $\Psi$ for the selected 10 QuickDraw categories, that demonstrate the discriminative power of our lattice-based graphical encoder. Intra-category instances tend to cluster together, suggesting category-level discriminative ability.

scalability and robustness of our latticed representation. Figure 5 shows some examples of sketch-to-sketch retrieval.

Visualization of $\Psi$ To further visualize the discriminative power of our lattice-based graphical encoder, we randomly select 100 sketches from each class in the test set, and visualize their latent vectors $\Psi$ using t-SNE in Figure 6. We observe that intra-category instances tend to cluster together, suggesting a category-level discriminative ability.

Ablation Study A thorough ablative study using sketch recognition accuracy, is conducted to verify the effectiveness of our different design choices, such as (i) the size of lattice graph, (ii) proximity principle for graph construction, and (iii) importance of residual connection used in our graphical sketch encoder. As shown in Table 2, we can observe that (i) increasing the value of $n$, that will simultaneously increase density of sampling lattice points from $S_L$, directly transpires in an improvement of recognition accuracy. From $n > 32$ we start to observe saturation of performance, thereby indicating the optimal value for $n = 32$. (ii) For construction of graph, we observe that more neighbors (nearby) works better than using only the nearest one for graph construction. (iii) removing residual connection leads to significant drop in performance, thereby establishing its importance.

On Complex Sketches To testify the robustness and applicability of our proposed latticed representation, we further investigate the performance when dealing with complex data. Specifically, we examine the recognition accuracy of the most complicated sketches (top 25%) over all categories, based on the number of strokes. From the results shown in Table 3, we can see that ours outperforms other competitors when healing the most complex sketches.
Human Study  To gain more insights about the fidelity of the healed sketches, a human study is additionally conducted. We recruited 10 participants. 50 sketch samples across all 10 classes were randomly selected. Each sample has two corrupted instances associated, at mask ratio 10% and 30%, respectively. For each corrupted sketch, we generate a group of healed sketches using different methods (SketchHealer, SketchRNN, SketchPix2seq, and ours). We show each participant, the corrupted sketch, and the group of four healed versions in random order. Each participant is then asked to pick a healed sketch that best resembles the corrupted input. Results in Table 4 reveals that according to human, sketches healed by our method resemble the corrupted input the best, at both corruption levels.

Table 4. Human study on fidelity of healed sketches (in %).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>5.80</td>
<td>11.55</td>
<td>38.98</td>
<td>43.67</td>
</tr>
<tr>
<td>30%</td>
<td>1.46</td>
<td>7.60</td>
<td>26.70</td>
<td>64.24</td>
</tr>
</tbody>
</table>

4.2. Image-to-Sketch Synthesis

Our Lattice-GCN-LSTM network can be applied to image-to-sketch translation. Essentially, given an input image, the corresponding edges are extracted using an off-the-shelf edge extractor [35]. Next, we transform the edge map into a latticed sketch representation using our lattice graph. The resulting SketchLattice could be seamlessly encoded via our graphical encoder. Finally, a vector sketch can be produced using the generative LSTM decoder. Our objective is to generate a sketch that best resembles the ground-truth sketch drawn by humans. Once trained, for any input image, we can obtain some representative lattice points based on the corresponding edges and lattice graph. Then, our generative LSTM decoder can generate a sketch from the sketch-level encoded representation $\Psi$, as stated in section 3.4.

Experimental Settings  We use QMUL-shoe-v2 [32], a fine-grained sketch-based image retrieval dataset, to evaluate our image-to-sketch synthesis approach. In total, there are 6648 one-to-one mappings of image-to-sketch pairs. The dataset is split into two parts, i.e., 6000 pairs for training and the rest of 648 pairs for testing. We chose the value of $n$ for our lattice graph as 32 and adopt the “nearby” strategy for graph construction. LS-SCC [23], a current state-of-the-art, is adopted for comparison.

Human Study  We conduct a user study that judges two aspects: (i) reality of the generated sketches, i.e., whether a sketch “looks” more like a human drawing (REAL), and (ii) similarity between the produced sketch and its target photo. Specifically, we show triplets of images, i.e., a photo shoe, and two corresponding sketches generated by LS-SCC [23] and ours in random order to 10 new participants. Each participant is asked to, (i) choose which of the two sketches “looks” more like a human drawing (REAL), and (ii) identify the sketch that best resembles the photo shoe (SIM).

Results and Analysis  Some qualitative results are shown in Figure 7, where we can see that results from both LS-SCC [23] and ours are far from satisfactory when compared to the human-drawn sketches, yet our generated sketches depict more detailed features, such as the “heel”, “sole” and the “zipper”. The human study results in Table 5 suggest that the produced sketches by our model are closer to human drawing, while equally effective to depict real shoes compared to LS-SCC.

Table 5. Human study on reality (REAL %) and similarity (SIM %) of the generated sketches.

<table>
<thead>
<tr>
<th>Method</th>
<th>REAL</th>
<th>SIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS-SCC [23]</td>
<td>44.82</td>
<td>49.77</td>
</tr>
<tr>
<td>Ours</td>
<td>55.18</td>
<td>50.23</td>
</tr>
</tbody>
</table>

5. Conclusion

We introduced a novel sketch representation, SketchLattice, that not only removes the bottleneck on having vector data, but also preserves the essential structural cues that vector data provides. This result in a sketch representation that is particularly amenable to structural changes that allows better abstraction modeling. We show this new representation helps multiple sketch manipulation tasks, such as sketch healing and image-to-sketch synthesis, where it outperforms state-of-the-art alternatives despite using significantly less parameters.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (NSFC) under 61601042.