User-Guided Line Art Flat Filling with Split Filling Mechanism

Lvmin Zhang
Soochow University / Style2Paints Research
China
lvminzhang@acm.org

Chengze Li
The Chinese University of Hong Kong / Style2Paints Research
China
ljsabc@gmail.com

Edgar Simo-Serra
Waseda University / JST PRESTO
Japan
ess@waseda.jp

Yi Ji
Soochow University
China
jiyi@suda.edu.cn

Tien-Tsin Wong
The Chinese University of Hong Kong
China
ttwong@cse.cuhk.edu.hk

Chunping Liu
Soochow University
China
cpliu@suda.edu.cn

Abstract

Flat filling is a critical step in digital artistic content creation with the objective of filling line arts with flat colors. We present a deep learning framework for user-guided line art flat filling that can compute the "influence areas" of the user color scribbles, i.e., the areas where the user scribbles should propagate and influence. This framework explicitly controls such scribble influence areas for artists to manipulate the colors of image details and avoid color leakage/contamination between scribbles, and simultaneously, leverages data-driven color generation to facilitate content creation. This framework is based on a Split Filling Mechanism (SFM), which first splits the user scribbles into individual groups and then independently processes the colors and influence areas of each group with a Convolutional Neural Network (CNN). Learned from more than a million illustrations, the framework can estimate the scribble influence areas in a content-aware manner, and can smartly generate visually pleasing colors to assist the daily works of artists. We show that our proposed framework is easy to use, allowing even amateurs to obtain professional-quality results on a wide variety of line arts.

1. Introduction

Flat filling is a process to color line arts with fairly flat colors according to the artists' specifications. This technique originates from the on-paper cartoon animation of the 1930s and remains critical in the digital painting era, as these flat-colored results exhibit great versatility in a wide variety of art workflows. Not only these flat colors can be directly blended with the line arts to create cartoon-like illustrations, they can also be used as independent foundations without blending the line arts for further adjustments towards more sophisticated and plentiful digital paintings.

In computer vision and graphics, two broad paradigms of user-guided line art colorization exist: traditional user scribble propagation and learning-based interactive colorization. In the first paradigm, typical methods like LazyBrush [28] and Manga Colorization [23] can explicitly and accurately control the "influence areas" of user scribbles, *i.e.*, the areas where those scribbles should propagate and influence, by matching the high-frequency/amplitude image constituents with hand-defined prior/energy. In professional use cases, the precise control of scribbles' influence areas is indispensable for artists in editing detailed colors. Ideally, supposing such influence areas are satisfactorily solved, the coloring procedure will be free from color leakages, as they only prop-

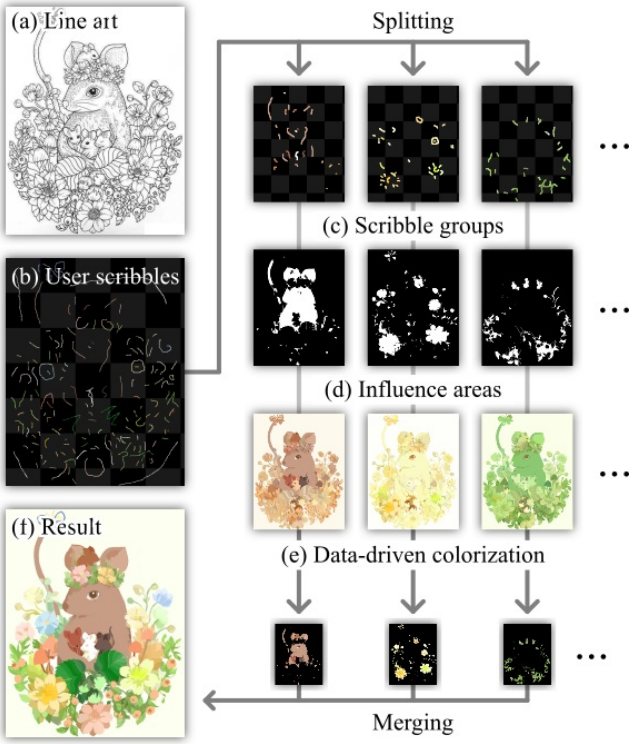


Figure 1. **Split Filling Mechanism (SFM).** Given (a) a line art and (b) some user scribbles, the SFM separate the scribbles into (c) several groups and estimate (d) the influence area of each group to accurately control the color segmentation. Afterwards, the SFM performs (e) data-driven colorization in each group to generate visually satisfying color combinations to assist artists. The outputs are merged together to achieve (f) the result. *Flower Mouse.*

agate color indices without color values contaminating each other. Besides, as scribble propagation entirely relies on user inputs, the related workflows are labor-intensive and require users to have artistic knowledge to obtain professional results.

In the second paradigm, typical learning-based interactive colorization methods such as PaintsChainer [29] and Zhang *et al*. [38] colorize line arts by learning parametric mappings from sparse lines and user inputs to colorful illustrations, and they can be post-processed with image flattening methods (*e.g*., [4, 37]) to meet the flat filling requirement. With the data-driven nature, these methods can "smartly" generate visually pleasing color combinations for in-the-wild line arts while reducing the burden on the artist and stimulating the artist's creation aspiration. Given that these methods do not explicitly control the influence areas of each scribble, users often need to go through tedious trail-and-errors when they attempt to control the accurate coloring on small detailed regions, or when they want to eliminate the color leakage between multiple adjacent scribbles.

Might we be able to get the best of both worlds, controlling the influence areas of user scribbles accurately to meet professional use cases, while at the same time incorporating data-driven color generation capability to inspire and facilitate content creation? We propose the Split Filling Mechanism (SFM) to achieve these two goals simultaneously as shown in Fig. 1. Firstly, to control the influence areas of scribbles, the SFM splits user scribbles (Fig. 1-(b)) into several groups (Fig. 1-(c)) and independently estimates the influence areas of each group (Fig. 1-(d)), preventing unwanted color contamination/leakage between scribble groups. Then, to generate colors for line arts through learning, the SFM framework learns from one million of illustrations to generate useful color combinations in each scribble group (Fig. 1-(e)). In this way, the split filling in these groups can be merged into the final output (Fig. 1-(f)), where the accurate control of scribble influence and the data-driven color generation capability is concurrently achieved, allowing for high quality line art flat filling.

Most creative tools in content manipulation are defined either non-parametrically, *e.g*., as an energy formulation, or with parametric mechanisms such as a deep learning model. Behaviors of those tools are therefore entirely conditioned by either human-defined propositions or machine-learned knowledge. Our approach differs in that it yields a joint effect of parametric models and non-parametric rules. Through SFM, the algorithm may end up with a more satisfactory procedure for interpreting user indications to flat filling results than what would be possible for either end-to-end learning models or human-defined principles.

Our contributions are as follows: (1) We analyze the merits and goals of traditional propagation-based and interactive learning-based colorization methods, and then motivate the problem to get the best of both worlds to simultaneously control the influence areas of user scribbles and generate plausible color combinations. (2) We propose the Split Filling Mechanism (SFM) framework consisting of the split scribble processing and data-driven colorization steps. (3) We show that the proposed approach can handle a diversity of complex line drawings, enabling both artists and amateurs to easily achieve high-quality flat filling results.

2. Related Work

Propagation-based line art colorization. Dating back to 1976 when the first flood-filling algorithm originated in the "bucket" tool on the first Apple computer, the Apple I [32], many other methods have also been proposed to fill colors within a user-defined area. Shaw *et al*. [25] proposes to speed up the flat colorization process using an efficient tree search algorithm. Optimization-based approaches have also been used to propagate the color from user scribbles in black-and-white photograph [17], and are further extended to the case of filling colors in patterned manga screen-tones [23], and many more. Among those approaches, *LazyBrush* [28] is one of the most well-known variants, which manages to fill

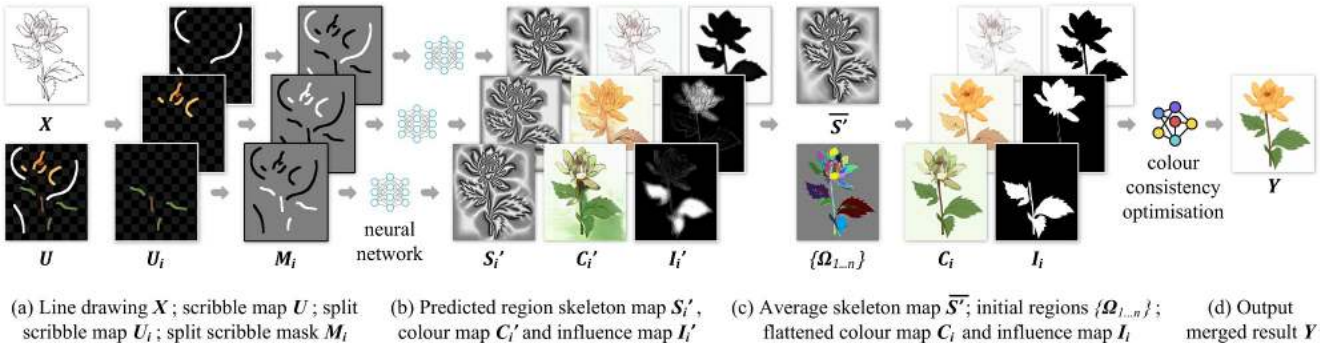


Figure 2. **Inference pipeline.** We visualize the components involved in the inference pipeline of our framework with splitting and merging.

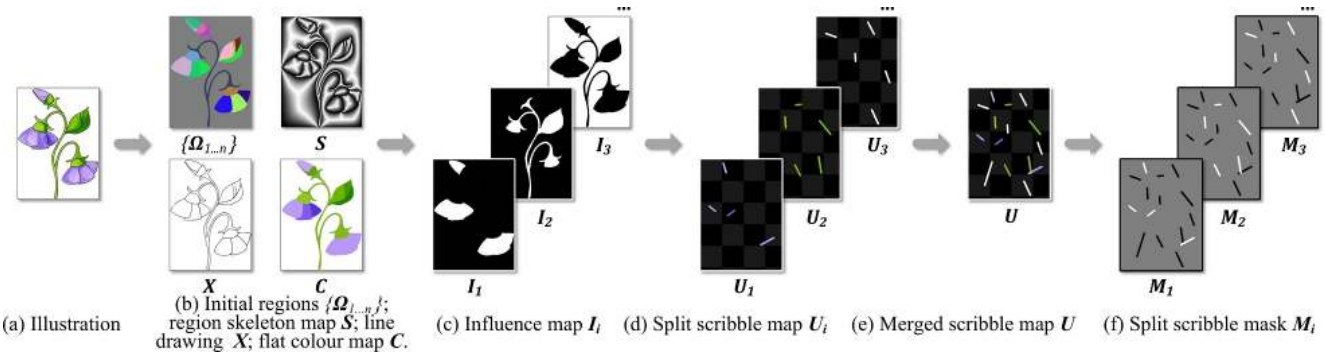


Figure 3. **Training data synthesis pipeline.** We illustrate the involved components within our dataset synthesis for splitting and merging.

flat colors in line drawings, rough sketches, and even screentoned manga. The popular open-source software GIMP uses an auto-closing algorithm [5] to compute user intended flat colorization regions in line drawings.

Learning-based line art colorization. The prosperity of large-scale learning techniques has popularized the initial researches of monochrome photography colorization [39, 12, 15, 40, 9, 36], and tailored research for illustration colorization followed up quickly. Scribbler [24] is proposed to colorize line drawings confining on bedroom scenes. Adversarial loss [8] has also been proven to be decisive in the line coloring due to their ability to produce more vivid and realistic colorizations, as shown in applications such as Deepcolor [6], Auto-painter [20], Tag2Pix [13], *etc*. Multiple solutions for direct sketch colorization have been proposed and widely used, including the commercial product PaintsChainer [29] and a two-stage solution [38]. Especially, [38] decomposes the colorization tasks into independent stages and achieves improved colorization quality. Besides learning a sketch-to-illustration mapping directly, style transfer can also be used for the line colorization tasks [19, 7, 11, 42, 10, 1]. The reference-based coloring can also work well on line video sequences, as proposed by [26] using an attention-based framework. Image stylization methods [2, 31, 33, 35, 34, 18, 30, 16] translate cartoon images or artistic drawings from/to photographs or human portraits.

Those approaches tend to produce results with pixel-level texture learned from their training data, whereas in many standard line-drawing-based artistic creation workflows (*e.g*., cel-coloring, cel-shading, *etc*.), artists have the requirement to fill color in regions, and the colors in each region must be flat and consistent. It remains a highly desired problem to ease the line art flat filling task with data-driven approaches.

3. Method

Overview. As shown in Fig. 2, the inputs of the framework are a gray scale line drawing $\boldsymbol{X} \in \mathbb{R}^{w \times h}$, with a width w and a height h, and an user scribble map $\boldsymbol{U} \in \mathbb{R}^{w \times h \times 4}$ with three RGB channels plus an alpha channel, whereas the output is a flat filling result $\boldsymbol{Y} \in \mathbb{R}^{w \times h \times 3}$. This framework first split the user scribbles $\boldsymbol{U}$ into N groups, yielding N split scribble maps $\boldsymbol{U}_i \in \mathbb{R}^{w \times h \times 4}$ with $i \in \{1, ..., N\}$ indicating the group index. The goal is to estimate the resulting color $\boldsymbol{C}_i \in \mathbb{R}^{w \times h \times 3}$ and influence area $\boldsymbol{I}_i \in \mathbb{R}^{w \times h}$ of each group, so as to merge them together to obtain the result $\boldsymbol{Y}$.

Splitting user scribbles. We cluster the colors used in the user scribble map $\boldsymbol{U}$ into N clusters using the k-means color clustering algorithm, and use the obtained color clusters to split the user scribble map $\boldsymbol{U}$ into a set of split scribble maps $\{\boldsymbol{U}_{1...N}\}$ (Fig. 2-(a)). One notice is that the naive RGB space k-means algorithm is relatively weak in differing colors with perceptually distinguishable minor hue/chroma difference,

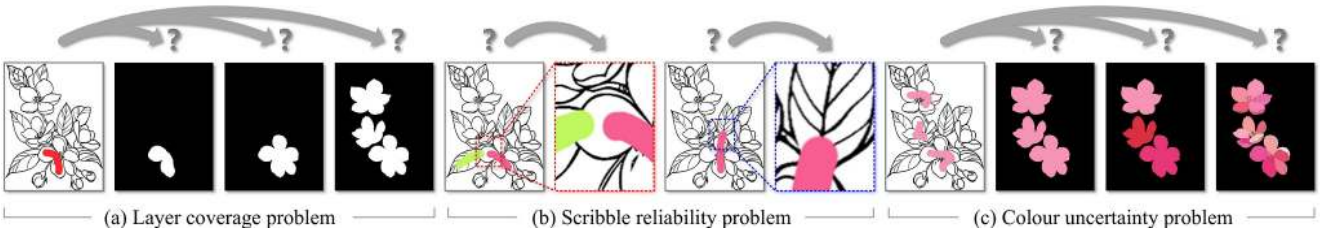


Figure 4. **Scribble problems.** We illustrate common problems associated with the user scribbles synthesizing steps that need to be resolved.

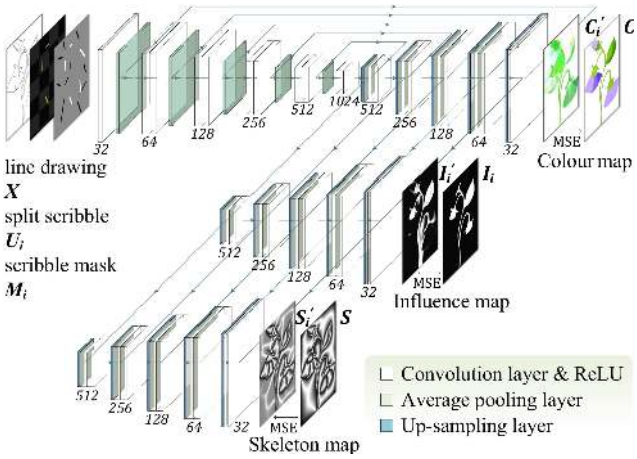


Figure 5. **Neural network architecture.** All convolutional layers use 3×3px kernels. We do not use any normalization layers. The Mean Squared Error is indicated as "MSE".

thus, we use a color chroma transform to enhance it as

$$[r, g, b]^\intercal \mapsto \left[\beta \cdot \frac{r+g+b}{3}, \frac{r}{r+g+b}, \frac{g}{r+g+b}\right]^\intercal . \quad (1)$$

Under ideal conditions, *i.e.*, if colors are fully separable with their intensity (axis 1), red chromaticity (axis 2), and green chromaticity (axis 3), this transform will perfectly separate different hue/chroma colors along the second two axes. While more sophisticated color spaces have been proposed (*e.g.*, [21]), we find that Eq. (1) is sufficient for the initial splitting. We scale the intensity by β to balance the importance of the intensity and chromaticity.

Masking scribbles. As shown in Fig. 2-(a), we propose to compute a scribble mask $\boldsymbol{M}_i$ to ease further learning tasks. In each mask $\boldsymbol{M}_i$, the scribble pixels in $\boldsymbol{U}_i$ are marked as "1", whereas the remaining scribble pixels in $\boldsymbol{U}$ are marked as "-1", and the other pixels are "0".

Estimating influence areas and resulting colors. We train a Convolutional Neural Network (CNN) to estimate the influence areas and resulting colors of each scribble group. As shown in Fig. 2-(b), the inputs of the neural network are the line drawing $\boldsymbol{X}$, split scribble map $\boldsymbol{U}_i$, and split scribble mask $\boldsymbol{M}_i$, whereas the outputs are the predicted region skeleton map $\boldsymbol{S}'_i$, color map $\boldsymbol{C}'_i$, and influence map $\boldsymbol{I}'_i$. These color and influence maps serve as a coarse initialization of the flat filling. The skeleton map is computed with Zhang and Suen 1984 [41]'s region skeleton intensity approach, which enables end-to-end region manipulation — arbitrary discrete regions can be converted to learnable per-pixel skeleton intensity with the *skeleton-from-region* transform (Appendix-A), and such skeleton can also reconstruct the original regions with the *region-from-skeleton* transform (Appendix-B).

Interpreting regions. As shown in Fig. 2-(c), we compute the average value of all estimated skeleton maps as $\overline{\boldsymbol{S'}} = \sum_{i=1}^{N} \boldsymbol{S}'_i / N$ and use the *region-from-skeleton* transform (Appendix-B) to obtain the regions $\{\boldsymbol{\Omega}_{i\ldots n}\}$. The final color map $\boldsymbol{C}_i$ is computed by filling all regions with the median colors sampled from the predicted color map $\boldsymbol{C}'_i$. Similarly, the final influence map $\boldsymbol{I}_i$ is computed from $\boldsymbol{I}'_i$ by setting "1" to the best i-th influence map with the largest value in each region, and "0" to the others.

Finalizing results. Finally, as shown in Fig. 2-(d), we perform a *color consistency optimization* in each color map $\boldsymbol{C}_i$ to merge adjacent regions selectively and replace several predicted colors with user scribble colors to improve the color consistency and tool usability. We use the same optimization approach in both the inference phase and later dataset synthesis. Afterwards, the final merging output is computed as

$$\boldsymbol{Y} = \sum_{i=1}^{N} \boldsymbol{C}_i \odot \boldsymbol{I}_i \quad . \quad (2)$$

where $\odot$ is the Hadamard product operator. We note that we apply the same weights to each RGB channel.

Color consistency optimization. The overall idea of this optimization is that we can merge several regions and their colors, so that the model can learn to smartly and selectively merge several regions and colorize them with consistent colors, according to the user scribbles and the input line drawing context and semantics. In particular, we note that each split scribble mask $\boldsymbol{M}_i$ indicates a region set Ψ_i as

$$\Psi_i = \{\boldsymbol{\Omega}_j \mid \exists p \in \boldsymbol{\Omega}_j, (\boldsymbol{M}_i)_p = 1\} \quad , \quad (3)$$

implying that the scribble mask $\boldsymbol{M}_i$ value is "1" for at least one pixel position p in a sampled region $\boldsymbol{\Omega}_j$, *i.e.*, at least one user scribble is located in the region $\boldsymbol{\Omega}_j$. Afterwards, we

estimate the set of region pairs $(\boldsymbol{\Omega}_a, \boldsymbol{\Omega}_b)$ that can be merged

$$\{(\boldsymbol{\Omega}_a, \boldsymbol{\Omega}_b) | \boldsymbol{\Omega}_a \in \Psi_i, \boldsymbol{\Omega}_b \in \mathcal{N}(\boldsymbol{\Omega}_a) \cap \Psi_i, ||\overline{\boldsymbol{\Omega}}_a - \overline{\boldsymbol{\Omega}}_b||_2 < \tau\}, \tag{4}$$

where $\mathcal{N}(\boldsymbol{\Omega}_i)$ indicates the set of neighbor regions to $\boldsymbol{\Omega}_i$, the term $\overline{\boldsymbol{\Omega}_i}$ is the mean color value of $\boldsymbol{\Omega}_i$, the operator $||\cdot||_2$ is the Euclidean distance, and τ is a threshold hyper-parameter. The pair set $\{(\boldsymbol{\Omega}_a, \boldsymbol{\Omega}_b)\}$ can be solved by brute force search, and we provide customized search steps and detailed replication guidelines in the supplementary material. We merge the region pairs from this set and replace their colors with the colors of the scribbles covering them to ensure the color consistency.

Dataset synthesis. As shown in Fig. 3, we synthesize a dataset to train our model with the paired data of the line drawing map $\boldsymbol{X}$, split scribble map $\boldsymbol{U}_i$, split scribble mask $\boldsymbol{M}_i$, skeleton map $\boldsymbol{S}$, color map $\boldsymbol{C}$, and the influence map $\boldsymbol{I}_i$. The $\{\boldsymbol{X}, \boldsymbol{U}_i, \boldsymbol{M}_i\}$ are the fed inputs, while the $\{\boldsymbol{S}, \boldsymbol{C}, \boldsymbol{I}_i\}$ are the learning objectives. To be specific, we sample one million illustrations in the Danbooru dataset [3]. Afterwards, given each illustration, we generate a line drawing map $\boldsymbol{X}$ with [27] (Fig. 3-(b)), and extract the initial regions $\{\boldsymbol{\Omega}_{i...n}\}$ with [37] (Fig. 3-(b)). Then, using the *skeleton-from-region* transform (Appendix-A), we convert these regions to a skeleton map $\boldsymbol{S}$ (Fig. 3-(b)). In parallel, by filling all regions with the median colors sampled from the illustration, we obtain the flat color map $\boldsymbol{C}$ (Fig. 3-(b)). Next, we use k-means in the aforementioned space (Eq. (1)) to cluster the flat color map $\boldsymbol{C}$ and obtain the influence maps $\boldsymbol{I}_i$ (Fig. 3-(c)). For each influence map $\boldsymbol{I}_i$, we synthesize a split scribble map $\boldsymbol{U}_i$ (Fig. 3-(d)), and these split scribble maps are merged into one scribble map $\boldsymbol{U}$ (Fig. 3-(e)). Finally, for each split scribble map $\boldsymbol{U}_i$, we compute the scribble mask $\boldsymbol{M}_i$ (Fig. 3-(f)).

User scribble simulation. To simulate each split scribble map $\boldsymbol{U}_i$, we use straight lines with 3px widths between two points $p_1, p_2 \in \mathbb{R}^2$. The scribble color is taken from the value of $\boldsymbol{C}$ at p_1, and the endpoints $\{p_1, p_2\}$ are randomly taken from a random region $\boldsymbol{\Omega}_j$ that belongs to a region set Φ_i specified by the current i-th influence map $\boldsymbol{I}_i$ with

$$\Phi_i = \{\boldsymbol{\Omega}_j \mid \forall p \in \boldsymbol{\Omega}_j, (\boldsymbol{I}_i)_p = 1\}\ , \tag{5}$$

indicating that the influence map $\boldsymbol{I}_i$ value is "1" for all pixel position p in the sampled region $\boldsymbol{\Omega}_j$. Next, we observe several common user scribbles as shown in Fig. 4 and note three typical problems. (1) *Layer coverage*. It is not clear what regions of the line drawing should be influenced by each scribble. As shown in Fig. 4-(a), scribbles may be needed to propagate to only nearby regions, or farther regions depending on the context and semantics. (2) *Scribble reliability*. Users in general will not provide strictly accurate scribbles, and instead may use conflicting colors for a single region, or scribbles that go past region boundaries as shown in Fig. 4-(b). (3) *Color uncertainty*. It is not clear how the user input scribble colors should be propagated. As shown in Fig. 4-(c), in some cases it may be important to give a flat color to the covered regions, while in others it may be preferable to generate color variations and transitions.

Figure 6. **Qualitative results with relatively simple line arts.** We present results with relatively simple and few scribbles. *Wolf, Jurassic, Tree Roots, Baby, Peony, and Mammoth.*

Although the SFM framework naturally mitigates these problems, care must be taken when synthesizing the training scribbles. In order to deal with the layer coverage issue, we randomly manipulate the region coverage of each scribble, so that the model can learn to estimate appropriate regions that are affected by each scribble. To be specific, instead of sampling from the same region $\boldsymbol{\Omega}_j$ for p_1 and p_2, we allow p_2 to be taken from a region that is reachable from p_1 within the region set Φ_i. We implement this by performing a random walk from $\boldsymbol{\Omega}_j$ to find a random k-step-neighbor region $\boldsymbol{\Omega}_k$ to sample p_2 from. We do a $k = 3$ step random walk to not obtain regions that are too far away. Next, to tackle the scribble reliability problem, we not only sample scribble endpoint positions within fixed region area, but also from a surrounding area with r pixel radius around the region (we use $r = 15$ by default) to simulate the coarse scribbles outside of the sampled regions. Finally, we perform the aforementioned *color consistency optimization* in each color map $\boldsymbol{C}_i$ to simulate color uncertainty, mimicking real scribbles that are coarsely drawn by artists.

Training. As shown in Fig. 5, we use a Fully Convolutional Neural Network (FCNN) with a common encoder and three decoders to predict a color map $\boldsymbol{C}'_i$, an influence map $\boldsymbol{I}'_i$, and a region skeleton map $\boldsymbol{S}'_i$, respectively. The loss function

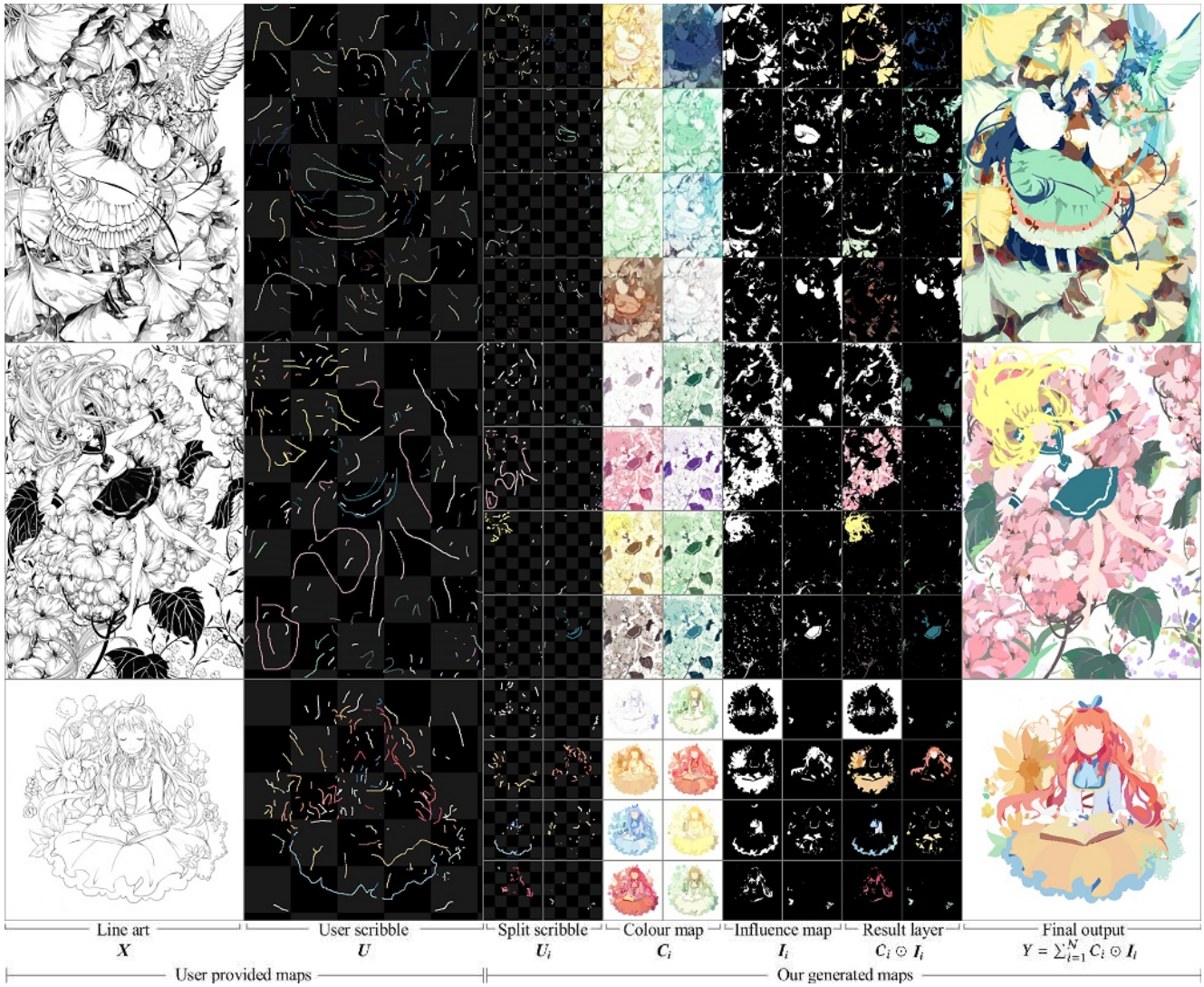


Figure 7. **Qualitative results with relatively complicated line arts.** We show a break-down of several results with our proposed approach. More examples are provided in the supplementary material. *Pea Princess, Flower with Alice, and Book Girl* © *used with artist permission.*

can be written as

$$L = \underbrace{\|\boldsymbol{C}'_i - \boldsymbol{C}\|_2^2}_{\text{Color maps}} + \underbrace{\|\boldsymbol{I}'_i - \boldsymbol{I}_i\|_2^2}_{\text{Influence maps}} + \underbrace{\|\boldsymbol{S}'_i - \boldsymbol{S}\|_2^2}_{\text{Skeleton maps}}, \quad (6)$$

where $\boldsymbol{C}$ is the ground truth color map, $\boldsymbol{I}_i$ is the ground truth influence map, and $\boldsymbol{S}$ is the ground truth region skeleton. It is notable that we neither use masked loss nor adversarial learning. Give that the architecture is fully convolutional, this model is applicable to images of adjustable resolutions.

4. Evaluation

4.1. Experimental setting

Compared approaches. We test several typical flat filling methods (or method combinations) of the traditional optimization-based scribble propagation method (1) LazyBrush [28], (2) Qu 2006 *et al*. [23], state-of-the-art deep learning sketch colorization method (3) Zhang *et al*. [38], method combinations (4) Zhang *et al*. [38] + GIMP region flattening [5] and (5) our framework.

Implementation details. The model is trained using the Adam optimiser [14] with a learning rate of $lr = 10^{-5}$, $\beta = 0.5$, a batch size of 16, and 20 epochs. Training is done with samples that are randomly cropped to be 224×224 pixels. LazyBrush [28] is tested with their official software. Qu 2006 *et al*. [23] is implemented with official parameters in their original paper. Zhang *et al*. [38] is the tested with the official and up-to-date software Style2Paints V4.5. GIMP region flattening [5] is from the software GIMP 2.10.22.

Hyper-parameters. We use the default (and recommended) configuration: $\beta = 0.5, N = 8$, and $\tau = 0.1$.

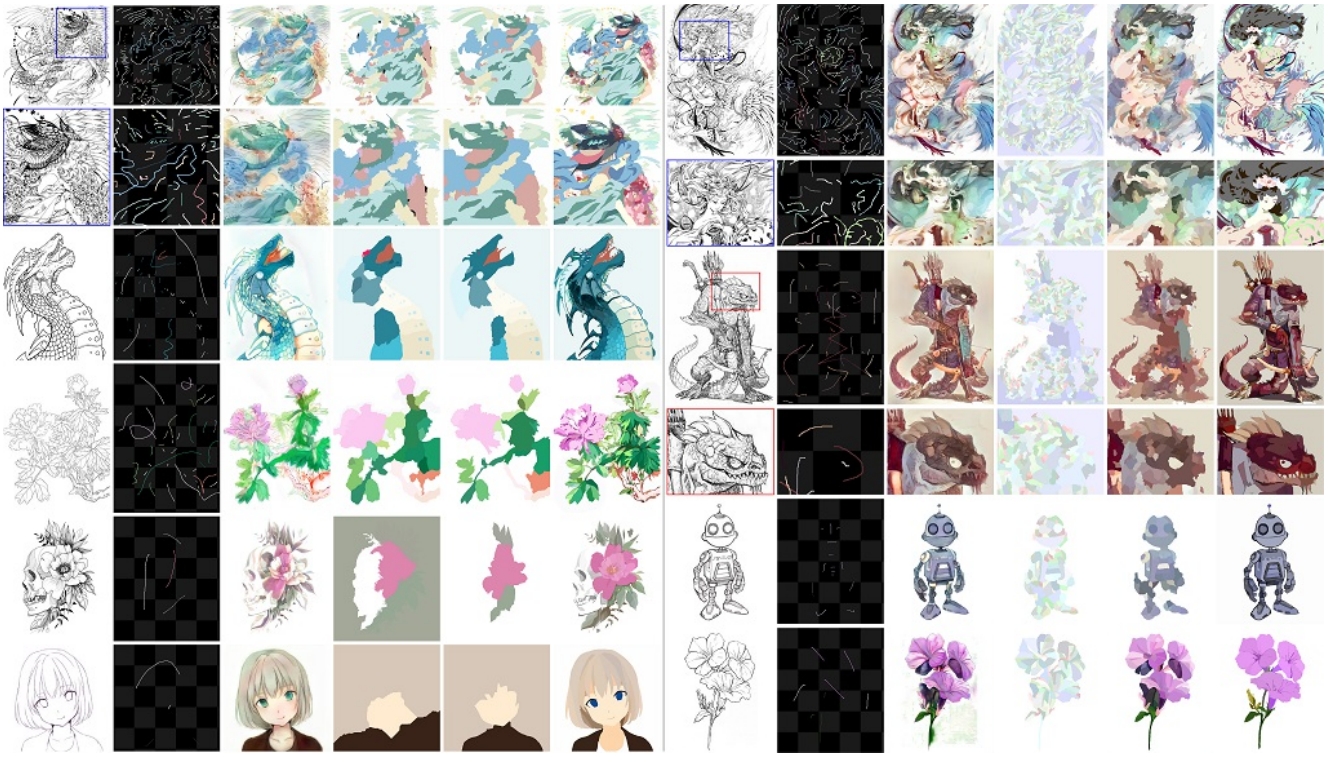


Figure 8. **Comparison with existing colorization methods.** Zoom in to see details. We compare our framework with [38, 23, 28] and the colorization method [38] combined with the segmentation method [5]. *Left: Flower Angel, Comollon, Wisteria Flowers, Poison Skull, and Megumi; Right: Fountain Angel, Goblin, Robot, and Azalea* © *used with artist permission.*

Figure	Auto region	Semi-auto region	Manual region
Flower Mouse	513 (46.43%)	510 (46.12%)	82 (7.44%)
Pea Princess	478 (47.31%)	442 (43.73%)	91 (8.96%)
Sky with Alice	551 (55.25%)	315 (31.56%)	132 (13.19%)
Prayer	530 (60.69%)	284 (32.52%)	59 (6.79%)
Flower with Alice	447 (54.56%)	261 (31.83%)	112 (13.62%)
Tree Elves	599 (59.13%)	366 (36.18%)	48 (4.69%)
Anna in Dream	643 (61.68%)	317 (30.42%)	82 (7.90%)
Book Girl	589 (53.80%)	377 (34.45%)	129 (11.75%)
Reading Awake	396 (44.61%)	362 (40.79%)	130 (14.60%)
Overall	53.72% $\pm$ 5.98%	36.40% $\pm$ 5.44%	9.88% $\pm$ 3.30%

Table 1. **Scribble analysis.** We perform an analysis of the number of scribbles used for the flat filling of different line drawings.

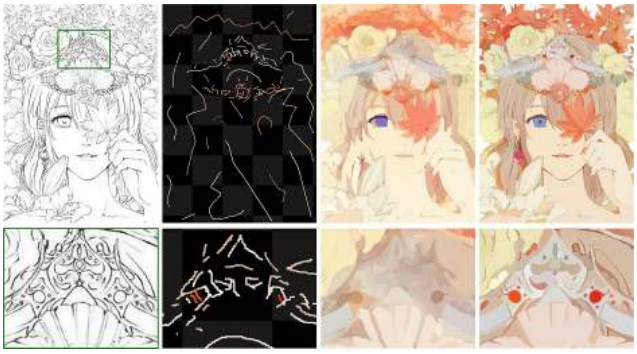
Testing samples. The tested images are Pixiv [22] and Danbooru [3] line arts, and original line arts from invited artists. We make sure that no similar line arts exists in the training dataset by removing the nearest line arts from the dataset with Mean Absolute Error (MAE) metric.

4.2. Qualitative results

Simple line arts. We first show some results on relatively simple line arts in Fig. 6. These results are obtained from non-artist amateur users with our framework. We can see that the line arts are flat-filled with plausible visual quality and can be directly used in many real-life artistic content creation workflows.

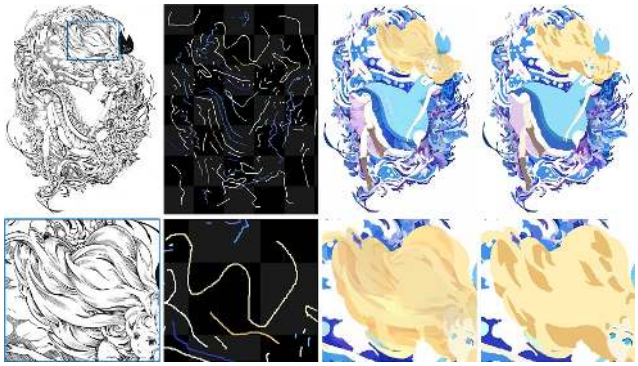
Complicated line arts. We show some qualitative results and layer break-downs of complicated line arts in Fig. 7. In particular, we visualize the input line are $\boldsymbol{X}$, scribble $\boldsymbol{U}$, and the output split scribble $\boldsymbol{U}_i$, color map $\boldsymbol{C}_i$, influence map $\boldsymbol{I}_i$, and the final result $\boldsymbol{Y}$. We can see how the proposed framework is applicable for line drawings with a large diversity and complexity to obtain high-quality results. More details and results are provided in the supplementary material.

Comparison to previous methods. We compare the proposed framework with existing deep learning and traditional algorithms in Fig. 8. We can see from the results that Zhang *et al*. [38] is unable to perform flat colorizations, while LazyBrush [28] and Qu 2006 *et al*. [23] are relatively weak in our specific illustration flat filling problem. The SFM approach is able to produce a detailed segmentation with satisfying flat coloring. We also compare our proposed approach with method combinations. Despite using the combination of two leading algorithms Zhang *et al*. [38] and GIMP region flattening [5], the obtained results are not as convincing as our proposed approach. We hypothesis that this is due to the fact we are not only learning to colorize, but also learning to perform a region segmentation and merging



Line art Scribble W/o SFM W/ SFM

Figure 9. **Significance of Split Filling Mechanism (SFM).** We compare the results obtained from our neural architecture with (w/) or without (w/o) the SFM. *One Leaf Knows Autumn* © *used with artist permission.*



Line art Scribble $\tau = 0.01$ $\tau = 0.1$

Figure 10. **Analysis of the τ parameter.** The parameter τ controls how the framework merges adjacent regions. For higher values of τ, more regions will be merged. Different use cases may need different values of τ. *Alice's Night* © *used with artist permission.*

that improves the flat filling jointly.

4.3. User study

Participants. We first test this framework with 9 non-artist amateur users and 3 professional artists. Those amateurs are college students without artistic knowledge.

Setups. As we found that the amateur users may have trouble with picking appropriate colors, we have added auxiliary color tables purchased from a professional cartoon studio with some examples shown in the supplementary material. By browsing the color tables, which will be made publicly available, amateur users are able to get quick inspiration for their scribble colors.

User guidelines. The users are asked to try best to color the given line arts. We do not give hard requirements and users can paint according to their perception.

Evaluation metrics. We calculate how many regions are filled manually by user scribbles, or filled automatically by our framework. In particular, we divide all colorized regions into three categories: (1) *Automatic regions.* (2) *Manual regions.* (3) *Semi-automatic regions.* Please see supplementary materials for detailed descriptions.

Results. As reported in Table 1, we can see that most of the regions are automatically or semi-automatically colored, with only roughly 10% of the regions being manually colored. This highlights how satisfactory results can be obtained with a relatively small amount of effort.

4.4. Ablative study

Significance of Split Filling Mechanism (SFM). We compare our framework with a cloned version but without the SFM processing. To be specific, we train our neural architecture (Fig. 5) with non-split data to directly estimate the final coloring and regions. In this setting, the training inputs become the line drawing $\boldsymbol{X}$ and non-split original user scribble map $\boldsymbol{U}$, whereas the outputs are the region skeleton map $\boldsymbol{S}'$ and color map $\boldsymbol{C}'$. We directly use the estimated $\boldsymbol{S}'$ to compute regions and flatten $\boldsymbol{C}'$ to get the final flat filling. All parameters and pipelines, including the color consistency optimization, remain same. The results are shown in Fig. 9. We can see that the SFM processing is an indispensable part of our framework. In absence of this processing, the outputs degenerate significantly yielding hardly usable flat filling.

Influence of parameters. As shown in Fig. 10, larger values of τ lead to more regions being merged, while low values can conserve too many details and lead to non-flat fillings.

5. Conclusion

We present a deep learning framework for user-guided line art flat filling. This framework can explicitly compute the influence areas of user scribbles, enabling users to edit the colors of image details and eliminate color leakage/contamination between scribbles. Different from traditional hand-crafted filling algorithms, the Split Filling Mechanism (SFM) is proposed to directly estimates the result colors and influence areas of scribbles, learned from a million illustrations. Results show that our SFM-based flat filling framework is able to handle diverse contents with complicated patterns and obtain reliable high-quality colorizations.

6. Acknowledgement

This technique is presented by Style2Paints Research. This work is supported by National Natural Science Foundation of China Nos 61972059, 61773272, 61602332; Natural Science Foundation of the Jiangsu Higher Education Institutions of China No 19KJA230001, Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University No93K172016K08; the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD). This work is also supported by JST PRESTO (Simo-Serra, Grant Number: JP-MJPR1756).

References

[1] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, volume 1, page 4, 2017.

[2] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. CartoonGAN: Generative adversarial networks for photo cartoonization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018.

[3] DanbooruCommunity. Danbooru2017: A large-scale crowd-sourced and tagged anime illustration dataset, 2018.

[4] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004.

[5] Sbastien Fourey, David Tschumperle, and David Revoy. A fast and efficient semi-guided algorithm for flat coloring line-arts. *EUROGRAPHICS*, 2018.

[6] Kevin Frans. Outline colorization through tandem adversarial networks. *In Arxiv*, 2017.

[7] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016.

[8] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *NIPS*, 3:2672–2680, 2014.

[9] Mingming He, Dongdong Chen, Jing Liao, Pedro V Sander, and Lu Yuan. Deep exemplar-based colorization. *ACM Transactions on Graphics*, 2018.

[10] Mingming He, Jing Liao, Lu Yuan, and Pedro V Sander. Neural color transfer between images. *ArXiv*, 2017.

[11] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018.

[12] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics*, 35(4), 2016.

[13] Hyunsu Kim, Ho Young Jhoo, Eunhyeok Park, and Sungjoo Yoo. Tag2pix: Line art colorization using text tag with secat and changing loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014.

[15] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, pages 577–593. Springer, 2016.

[16] Junsoo Lee, Eungyeup Kim, Yunsung Lee, Dongjun Kim, Jaehyuk Chang, and Jaegul Choo. Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[17] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM Transactions on Graphics*, 2004.

[18] Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. Im2pencil: Controllable pencil illustration from photographs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[19] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Transactions on Graphics*, 36(4):1–15, jul 2017.

[20] Yifan Liu, Zengchang Qin, Zhenbo Luo, and Hua Wang. Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks. *In Arxiv*, 2017.

[21] I. Omer and M. Werman. Color lines: image specific color representation. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* IEEE, 2004.

[22] pixiv.net. pixiv. *pixiv*, 2007.

[23] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Transactions on Graphics*, 25(3):1214–1220, July 2006.

[24] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. *CVPR*, 2017.

[25] John R. Shaw. Quickfill: An efficient flood fill algorithm. *codeproject*, 2004.

[26] Min Shi, Jia-Qi Zhang, Shu-Yu Chen, Lin Gao, Yu-Kun Lai, and Fang-Lue Zhang. Deep line art video colorization with a few references. *Arxiv*, 2020.

[27] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering Sketching: Adversarial Augmentation for Structured Prediction. *ACM Transactions on Graphics*, 37(1), 2018.

[28] Daniel Sykora, John Dingliana, and Steven Collins. LazyBrush: Flexible painting tool for hand-drawn cartoons. *Computer Graphics Forum*, 28(2), 2009.

[29] TaiZan. Paintschainer tanpopo. *PreferredNetwork*, 2016.

[30] Matteo Tomei, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Art2Real: Unfolding the Reality of Artworks via Semantically-Aware Image-to-Image Translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[31] Xinrui Wang and Jinze Yu. Learning to cartoonize using white-box cartoon representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[32] Stephen Gary Wozniak. The apple i computer. *Apple*, 1976.

[33] Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L. Rosin. APDrawingGAN: Generating artistic portrait drawings from face photos with hierarchical GANs. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2019.

[34] Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. Unpaired portrait drawing generation via asymmetric cycle mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '20)*, 2020.

[35] Ran Yi, Mengfei Xia, Yong-Jin Liu, Yu-Kun Lai, and Paul L. Rosin. Line drawings for face portraits from photos using global and local structure based GANs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[36] B. Zhang, M. He, J. Liao, P. V. Sander, L. Yuan, A. Bermak, and D. Chen. Deep exemplar-based video colorization. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8044–8053, 2019.

[37] Lvmin Zhang, Yi JI, and Chunping Liu. Danbooregion: An illustration region dataset. In *European Conference on Computer Vision (ECCV)*, 2020.

[38] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. In *ACM Transactions on Graphics*, 2018.

[39] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[40] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics*, 9(4), 2017.

[41] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 1984.

[42] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NIPS*, 2017.