User-Guided Line Art Flat Filling with Split Filling Mechanism

LVMIN ZHANG, Soochow University / Style2Paints Research, China
CHENGZE LI, The Chinese University of Hong Kong / Style2Paints Research, China
EDGAR SIMO-SERRA, Waseda University / JST PRESTO, Japan
YI JI, Soochow University, China
TIEN-TSIN WONG, The Chinese University of Hong Kong, China
CHUNPING LIU, Soochow University, China

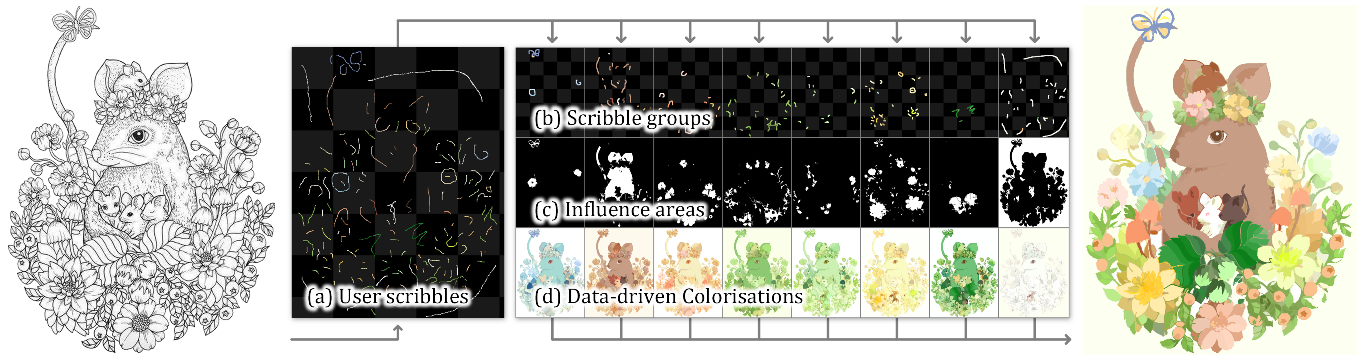


Fig. 1. **Split Filling Mechanism (SFM).** Given the left line art and (a) some user scribbles, the SFM separate the scribbles into (b) several groups and estimate (c) the influence area of each group to accurately control the colour segmentation. Afterwards, the SFM performs (d) data-driven colourisation in each group to generate visually satisfying colour combinations to assist artists. The outputs are merged together to achieve the right result. *Flower Mouse © used with artist permission.*

Flat filling is a critical step in digital artistic content creation with the objective of filling line arts with flat colours. We present a deep learning framework for user-guided line art flat filling that can compute the "influence areas" of the user colour 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 colours of image details and avoid colour leakage/contamination between scribbles, and simultaneously, leverages data-driven colour 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 colours 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 colours 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 colour line arts with fairly flat colours 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-coloured results exhibit great versatility in a wide variety of art workflows. Not only these flat colours 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 colourisation exist: traditional user scribble propagation and learning-based interactive colourisation. In the first paradigm, typical methods like LazyBrush [Sykora et al. 2009] and Manga Colourisation [Qu et al. 2006] 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 colours. Ideally, supposing such influence areas are satisfactorily solved, the colouring procedure will be free from colour leakages, as they only propagate colour indices without colour values contaminating each other. Besides, as scribble propagation entirely relies on user inputs, the related workflows are labour-intensive and require users to have artistic knowledge to obtain professional results.

In the second paradigm, typical learning-based interactive colourisation methods such as PaintsChainer [TaiZan 2016] and Zhang *et al.* [Zhang et al. 2018] colourise line arts by learning parametric mappings from sparse lines and user inputs to colourful illustrations, and they can be post-processed with image flattening methods (*e.g.*, [Felzenszwalb and Huttenlocher 2004; Zhang et al. 2020]) to meet the flat filling requirement. With the data-driven nature, these methods can "smartly" generate visually pleasing colour 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 colouring on small detailed regions, or when they want to eliminate the colour 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 colour 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-(a)) into several groups (Fig. 1-(b)) and independently estimates the influence areas of each group (Fig. 1-(c)), preventing unwanted colour contamination/leakage between scribble groups. Then, to generate colours for line arts through learning, the SFM framework learns from one million of illustrations to generate useful colour combinations in each scribble group (Fig. 1-(d)). In this way, the split filling in these groups can be merged into the final output (Fig. 1-right), where the accurate control of scribble influence and the data-driven colour 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. Behaviours 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 analyse the merits and goals of traditional propagation-based and interactive learning-based colourisation 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 colour combinations. (2) We propose the Split Filling Mechanism (SFM) framework consisting of the split scribble processing and data-driven colourisation 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

Flat Filling. Since the original "flood filling" algorithm, going back to the "bucket" tool on the first Apple computer, the Apple I [Wozniak 1976], many methods have been proposed to fill colours within a user-defined area. Shaw *et al.* [2004] propose to speed up the flat filling process using an efficient tree search algorithm. Optimisation-based approaches have also been used to propagate the colour from user scribbles in black-and-white photograph [Levin et al. 2004], and further extended to the case of filling colours in patterned manga screen-tones [Qu et al. 2006]. *LazyBrush* [Sykora et al. 2009] is one of the most well known variants of an optimisation approach to fill colour in line drawings, rough sketches, and even screen-toned manga. The popular opensource software GIMP uses an auto-closing algorithm [Fourey et al. 2018] to compute flat filling regions in line drawings. Trapped-ball flood filling [Hensman and Aizawa 2017] is also commonly used to avoid small gaps in the line drawings. Our proposed framework view the flat filling workflow as a split-and-merge problem, and uses a data-driven approach to learn the colourisation from one million illustrations.

Learning-based Sketch Colourisation. The prosperity of large-scale learning techniques with copious amounts of available illustrations has popularised the usage of neural networks to colourise images. Although initial research focused on black-and-white photography colourisation [He et al. 2018; Iizuka et al. 2016; Larsson et al. 2016; Zhang et al. 2016, 2017b], research on the colourisation of line drawings quickly appeared. Scribbler [Sangkloy et al. 2017] was proposed to colour different line drawings with a particular focus on bedroom scenes. Furusawa *et al.* [2017] proposed to colourise manga with given different colour palettes. Adversarial losses [Goodfellow et al. 2014] have also proved popular in approaches such as Deepcolour [Frans 2017] and Auto-painter [Liu et al. 2017] due to their ability to produce more vivid and realistic colourisations. PaintsChainer [TaiZan 2016] is an example of a commercial product for learning-based sketch colourisation. A two-stage framework to decompose the colourisation task into two independent stages was proposed in [Zhang et al. 2018]. More recently, an attention-based framework for line art video colourisation based on a few video frame references has been proposed [Shi et al. 2020]. As an alternative to directly learning a colourisation approach, style transfer has also been used for colourisation [Chen et al. 2017; Gatys et al. 2016; He et al. 2017; Hoffman et al. 2018; Liao et al. 2017; Zhu et al. 2017]. Related to colourisation of line drawings, learning-based inking approaches [Li et al. 2017; Liu et al. 2015; Simo-Serra et al. 2018a,b, 2016] can also be used to improve the input sketches and consequently the afterwards colourisation. It is notable that those approaches tend to produce results with pixel-level texture learnt from their training data, whereas in many standard line-drawing-based artistic creation workflows (*e.g.*, cel-colouring, cel-shading, *etc.*), artists need to fill colour in regions, and the colours in each region must be flat and consistent. Thus, it remains an open problem to ease the line drawing flat filling task using data-driven or learning-based approaches. In this research, we focus on the flat filling which plays a very important role in modern illustration and digital painting production.

Decomposition-based Image Processing. Decomposition is a unique paradigm in image processing in which the splitting allows easier manipulation of the image, and is especially used in colour layer editing and blending. One of the most famous approaches is intrinsic image decomposition [Barrow and Tenenbaum 1978], in which images are split into albedo and shading layers. Generally it is seen as a constrained optimisation problem [Shen et al. 2011], while recent approaches leverage learning-based algorithms to directly learn the mapping between input images and their albedo from large amounts of data [Barron and Malik 2012; Gehler et al. 2011; Serra et al. 2012]. Apart from intrinsic image decomposition, it is common to fully split images into several component layers for image editing

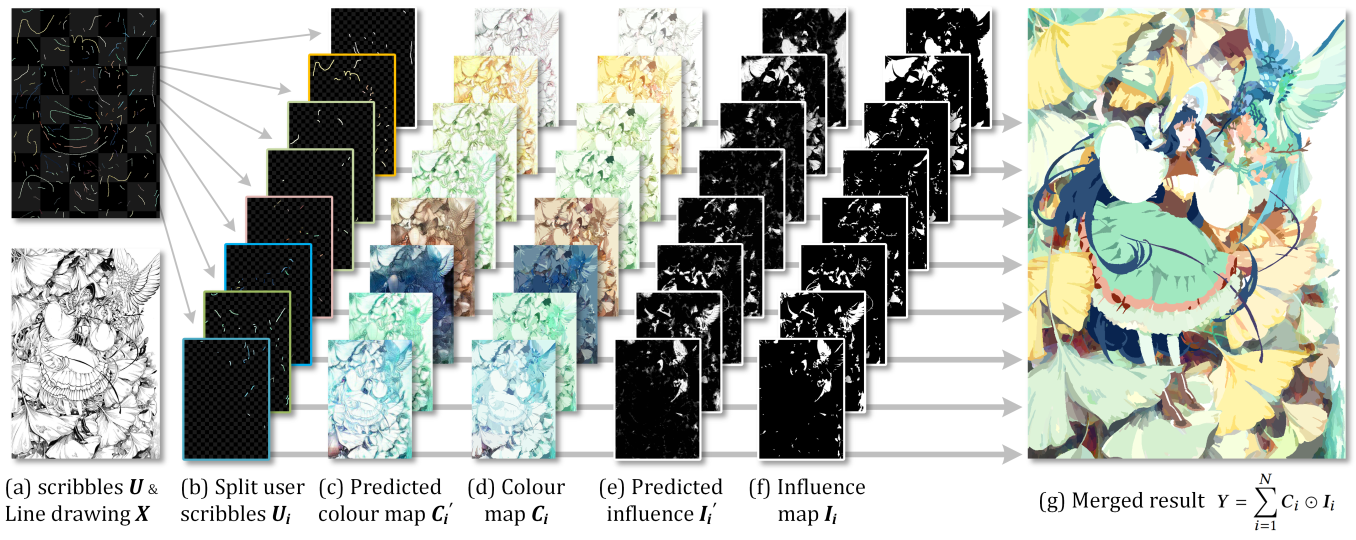

(a) scribbles $\boldsymbol{U}$ & Line drawing $\boldsymbol{X}$
(b) Split user scribbles $\boldsymbol{U_i}$
(c) Predicted colour map $\boldsymbol{C_i}'$
(d) Colour map $\boldsymbol{C_i}$
(e) Predicted influence $\boldsymbol{I_i}'$
(f) Influence map $\boldsymbol{I_i}$
(g) Merged result $Y = \sum_{i=1}^{N} C_i \odot \boldsymbol{I}_i$

Fig. 2. **Framework overview.** Our framework splits the user scribbles by colours into virtual palette cells, using each cell to colour the image independently as shown by each row. Finally, all the different colourisation results are merged into a single output image. *Pea Princess © used with artist permission.*

purposes before merging them back [Lin et al. 2017]. This decomposition can be done adaptively based on colour segmentation [Aksoy et al. 2017] or based on constrained optimisation [Koyama and Goto 2018]. Zhang *et al.* [2017a] propose to optimise the decomposition to re-colour images. In this paper, our framework leverages a split filling mechanism that groups the different user scribbles to produce different colourisations that are finally merged together into a single final result.

3 METHOD

3.1 Split filling mechanism

3.1.1 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 $Y \in \mathbb{R}^{w \times h \times 3}$. 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 $i \in \{1, \dots, N\}$ indicating the group index. The goal is to estimate the resulting colour $\boldsymbol{C}_i \in \mathbb{R}^{w \times h \times 3}$ 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 $\boldsymbol{Y}$.

3.1.2 Splitting user scribbles. We cluster the colours used in the user scribble map $\boldsymbol{U}$ into N clusters using the k-means colour clustering algorithm, and use the obtained colour clusters to split the user scribble map $\boldsymbol{U}$ into a set of split scribble maps $\{\boldsymbol{U}_{1 \dots N}\}$ (Fig. 3-(a)). One notice is that the naive RGB space k-means algorithm is relatively weak in differing colours with perceptually distinguishable minor hue/chroma difference, thus, we use a colour chroma transform to enhance it as

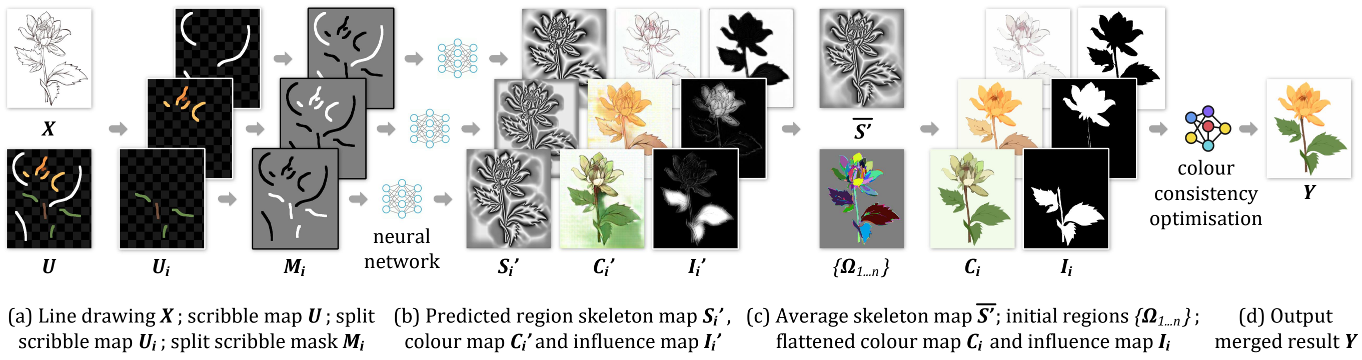
$$[r, g, b]^\intercal \mapsto \left[\beta \cdot \frac{r+g+b}{3}, \frac{r}{r+g+b}, \frac{g}{r+g+b} \right]^\intercal . \qquad (1)$$

Under ideal conditions, *i.e.*, if colours 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 colours along the second two axes. While more sophisticated colour spaces have been proposed (*e.g.*, [Omer and Werman 2004]), 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.

3.1.3 Masking scribbles. As shown in Fig. 3-(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".

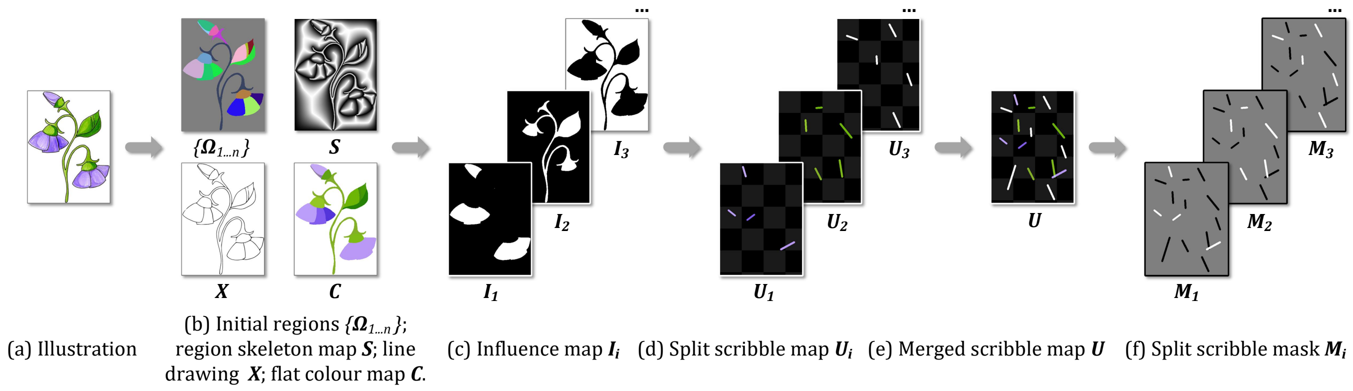
3.1.4 Estimating influence areas and resulting colours. We train a Convolutional Neural Network (CNN) to estimate the influence areas and resulting colours of each scribble group. As shown in Fig. 3-(b), the inputs of the neural network are the line drawing $\boldsymbol{X}$, split scribble map $\boldsymbol{U}_i$, and split scribble mask $\boldsymbol{M}_i$, whereas the outputs are the predicted region skeleton map $\boldsymbol{S}_i'$, colour map $\boldsymbol{C}_i'$, and influence map $\boldsymbol{I}_i'$. These colour and influence maps serve as a coarse initialization of the flat filling. The skeleton map is computed with Zhang and Suen 1984 [Zhang and Suen 1984]'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).

3.1.5 Interpreting regions. As shown in Fig. 3-(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 $\{\Omega_{i \dots n}\}$. The final colour map $\boldsymbol{C}_i$ is computed by filling all regions with the median colours sampled from the predicted colour 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.



X
U
U_i
M_i
neural network
S_i'
C_i'
I_i'
S̄'
{Ω_{1...n}}
C_i
I_i
colour consistency optimisation
Y
(a) Line drawing X ; scribble map U ; split scribble map U_i ; split scribble mask M_i
(b) Predicted region skeleton map S_i', colour map C_i' and influence map I_i'
(c) Average skeleton map S̄'; initial regions {Ω_1...n}; flattened colour map C_i and influence map I_i
(d) Output merged result Y

Fig. 3. **Inference pipeline.** We visualise the components involved in the inference pipeline of our framework. *Flower #1 (line art) © used with artist permission.*



{Ω_1...n}
S
X
C
I_1
I_2
I_3
U_1
U_2
U_3
U
M_1
M_2
M_3
(a) Illustration
(b) Initial regions {Ω_1...n}; region skeleton map S; line drawing X; flat colour map C.
(c) Influence map I_i
(d) Split scribble map U_i
(e) Merged scribble map U
(f) Split scribble mask M_i

Fig. 4. **Training data synthesis pipeline.** We illustrate the involved components within our dataset synthesis step. *Flower #2 © used with artist permission.*

3.1.6 *Finalizing results.* Finally, as shown in Fig. 3-(d), we perform a *colour consistency optimization* in each colour map C_i to merge adjacent regions selectively and replace several predicted colours with user scribble colours to improve the colour 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

$$Y = \sum_{i=1}^{N} C_i \odot \boldsymbol{I}_i \quad . \tag{2}$$

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

3.1.7 *colour consistency optimization.* The overall idea of this optimization is that we can merge several regions and their colours, so that the model can learn to smartly and selectively merge several regions and colourise them with consistent colours, 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

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

implying that the scribble mask $\boldsymbol{M}_i$ value is "1" for at least one pixel position p in a sampled region Ω_j, *i.e.*, at least one user scribble is located in the region Ω_j. Afterwards, we estimate the set of region pairs (Ω_a, Ω_b) that can be merged

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

where $\mathcal{N}(\Omega_i)$ indicates the set of neighbor regions to Ω_i, the term $\overline{\Omega}_i$ is the mean colour value of Ω_i, the operator $||\cdot||_2$ is the Euclidean distance, and τ is a threshold hyper-parameter. The pair set $\{(\Omega_a, \Omega_b)\}$ 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 colours with the colours of the scribbles covering them to ensure the colour consistency.

3.2 Data preparation and optimisation

3.2.1 *Dataset synthesis.* As shown in Fig. 4, 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 $\boldsymbol{S}$, colour 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 [DanbooruCommunity 2018]. Afterwards, given each illustration, we generate a line drawing map $\boldsymbol{X}$ with [Simo-Serra et al. 2018a] (Fig. 4-(b)), and extract the initial regions $\{\Omega_{i...n}\}$ with [Zhang et al. 2020] (Fig. 4-(b)). Then, using the *skeleton-from-region* transform (Appendix-A),

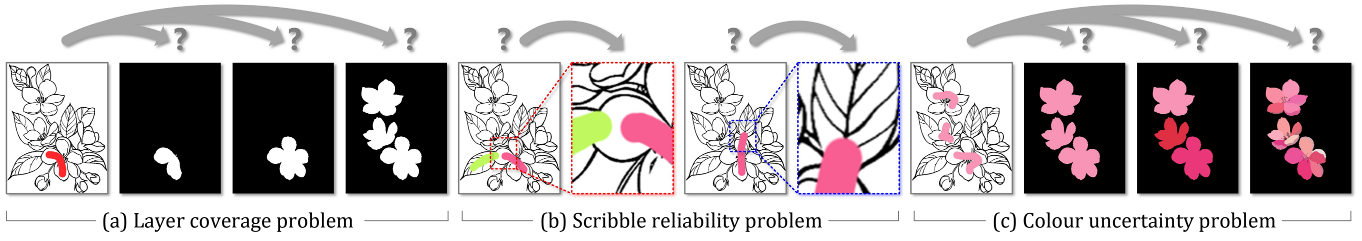

(a) Layer coverage problem
(b) Scribble reliability problem
(c) Colour uncertainty problem

Fig. 5. **Scribble problems.** We illustrate common problems associated with the user scribbles synthesising steps. Our scribble synthesising approach is tailored to resolve these problems. © *used with artist permission.*

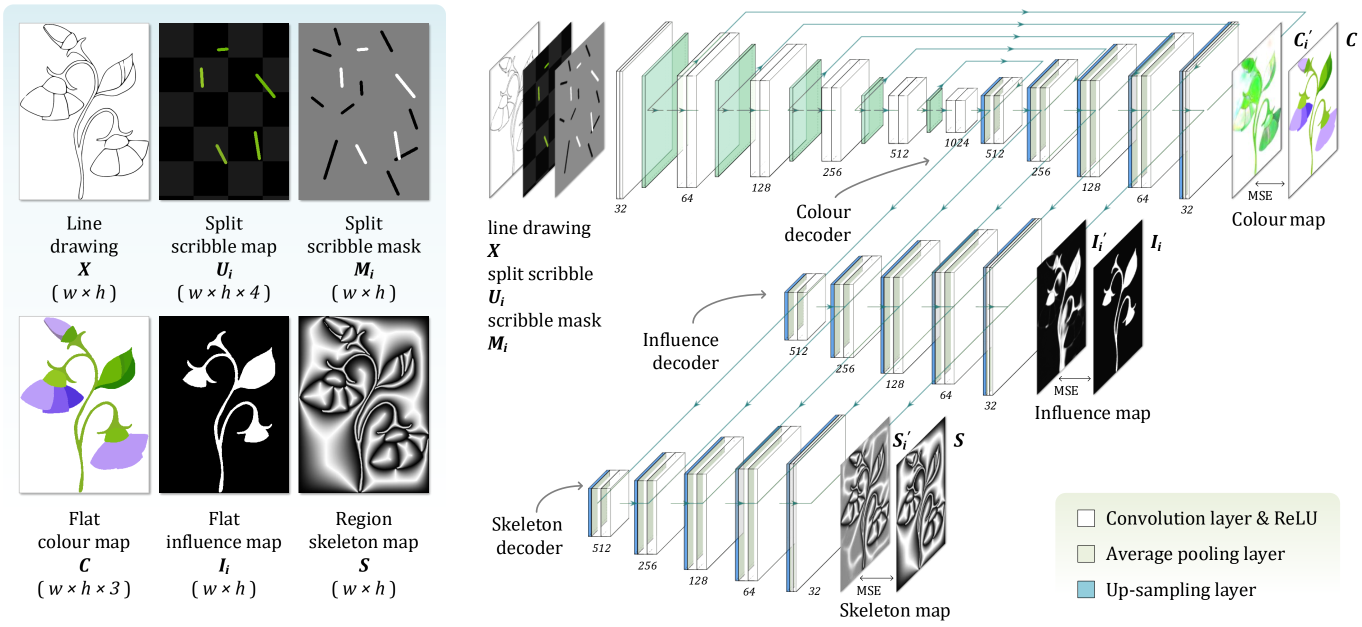


Fig. 6. **Neural network architecture.** On the left we visualise the different components, and on the right we visualise the full architecture. All convolutional layers use 3 × 3px kernels. We do not use any normalisation layers. The Mean Squared Error is indicated as "MSE".

we convert these regions to a skeleton map $\boldsymbol{S}$ (Fig. 4-(b)). In parallel, by filling all regions with the median colours sampled from the illustration, we obtain the flat colour map $\boldsymbol{C}$ (Fig. 4-(b)). Next, we use k-means in the aforementioned space (Eq. (1)) to cluster the flat colour map $\boldsymbol{C}$ and obtain the influence maps $\boldsymbol{I}_i$ (Fig. 4-(c)). For each influence map $\boldsymbol{I}_i$, we synthesize a split scribble map $\boldsymbol{U}_i$ (Fig. 4-(d)), and these split scribble maps are merged into one scribble map $\boldsymbol{U}$ (Fig. 4-(e)). Finally, for each split scribble map $\boldsymbol{U}_i$, we compute the scribble mask $\boldsymbol{M}_i$ (Fig. 4-(f)).

3.2.2 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 colour 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 Ω_j that belongs to a region set Φ_i specified by the current i-th influence map $\boldsymbol{I}_i$ with

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

indicating that the influence map $\boldsymbol{I}_i$ value is "1" for all pixel position p in the sampled region Ω_j. Next, we observe several common user scribbles as shown in Fig. 5 and note three typical problems. (i) *Layer coverage.* It is not clear what regions of the line drawing should be influenced by each scribble. As shown in Fig. 5-(a), scribbles may be needed to propagate to only nearby regions, or farther regions depending on the context and semantics. (ii) *Scribble reliability.* Users in general will not provide strictly accurate scribbles, and instead may use conflicting colours for a single region, or scribbles that go past region boundaries as shown in Fig. 5-(b). (iii) *colour uncertainty.* It is not clear how the user input scribble colours should be propagated. As shown in Fig. 5-(c), in some cases it may be important to give a flat colour to the covered regions, while in others it may be preferable to generate colour variations and transitions.

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

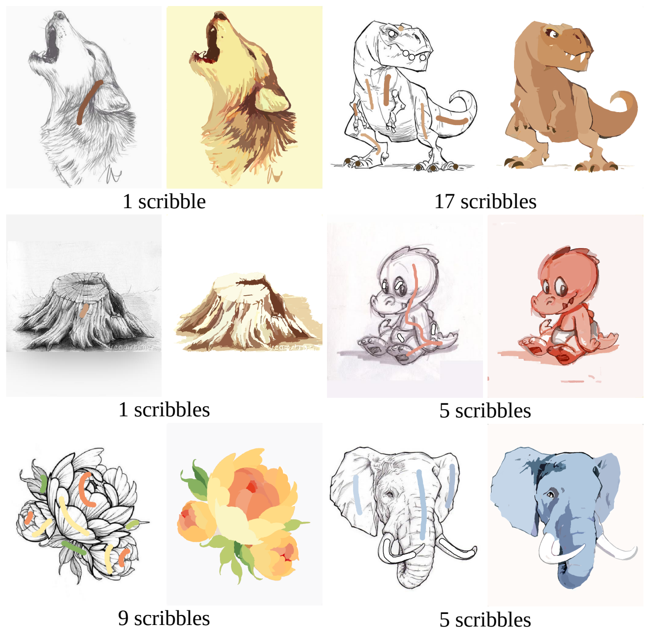


Fig. 7. **Working with a small number of scribbles.** We present results with relatively simple and fewer scribbles, *i.e.*, less than 20 scribbles. *Big Wolf, Jurassic, Old Tree Roots, Baby Dragon, Peony, and Mammoth © used with artist permission.*

estimate appropriate regions that are affected by each scribble. To be specific, instead of sampling from the same region Ω_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 Ω_j to find a random k-step-neighbor region Ω_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 *colour consistency optimization* in each colour map C_i to simulate colour uncertainty, mimicking real scribbles that are coarsely drawn by artists.

3.2.3 Training. As shown in Fig. 6, we use a Fully Convolutional Neural Network (FCNN) with a common encoder and three decoders to predict a colour map C'_i, an influence map I'_i, and a region skeleton map S'_i, respectively. The loss function can be written as

$$L = \underbrace{\|C'_i - C\|_2^2}_{\text{colour maps}} + \underbrace{\|I'_i - I_i\|_2^2}_{\text{Influence maps}} + \underbrace{\|S'_i - S\|_2^2}_{\text{Skeleton maps}} , \tag{6}$$

where C is the ground truth colour map, I_i is the ground truth influence map, and 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 EXPERIMENTS

In this section, we perform a set of comparisons with existing approaches and ablation studies for an in-depth evaluation of the proposed framework.

4.1 Examples with simple line art

We show some results with a relatively small number of scribbles in Fig. 7. These results are obtained from non-artist amateur users with our framework. All those results are achieved with less than 20 scribbles. Despite using a limited number of imprecise scribbles, the line drawings can still be flat-filled with plausible visual quality and can be directly used in many real-life artistic content creation workflows.

4.2 User study

We first conduct a user study to test this framework with non-artist amateur users. As we found that these users may have trouble with picking appropriate colours, we have added auxiliary colour tables purchased from a professional cartoon studio with some examples shown in Fig. 8. By browsing these tables, which will be made publicly available, amateur users are able to get quick inspiration for their scribble colours. A break-down of an example result is shown in Fig. 9. We can see how the amateur user can obtain near professional-quality results, despite having no experience in artistic illustration and digital painting. More examples are shown in the supplementary materiel, including results from both amateur users and professional artists.

4.3 Qualitative result

We show some qualitative results and layer break-downs in Fig. 10. 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.

4.4 Comparison with existing approach

We compare the proposed framework with existing deep learning and traditional algorithms in Fig. 11. In particular, we compare with Two-Stage Sketch Colourisation [Zhang et al. 2018], Manga colourisation [Qu et al. 2006], and LazyBrush [Sykora et al. 2009]. We can see from the results that the deep-learning-based Two-Stage Sketch Colourisation is unable to perform flat colourisation, while the optimisation-based approaches of Manga colourisation and LazyBrush are relatively weak in our specific illustration flat colourisation problem. Our split-and-merge approach is able to produce a detailed segmentation with satisfying flat colouring.

We also compare our proposed approach with a combination of existing approaches in Fig. 12. In particular, we use the GIMP line drawing region extraction [Fourey et al. 2018] to obtain a region segmentation, and use that to flatten the colours of the Two-Stage Sketch Colourisation method [Zhang et al. 2018]. Despite using the combination of two leading algorithms, the obtained results are not as convincing as our proposed approach. We hypothesise that this is due to the fact we are not only learning to colourise, but

Energetic
Romantic
Cute
Fresh
Elegant
Simple
Science-fiction

Fig. 8. **Auxiliary colour tables.** We have collected 851 professional colour tables to help amateur users selecting colours, which we will make publicly available. *© used with artist permission.*

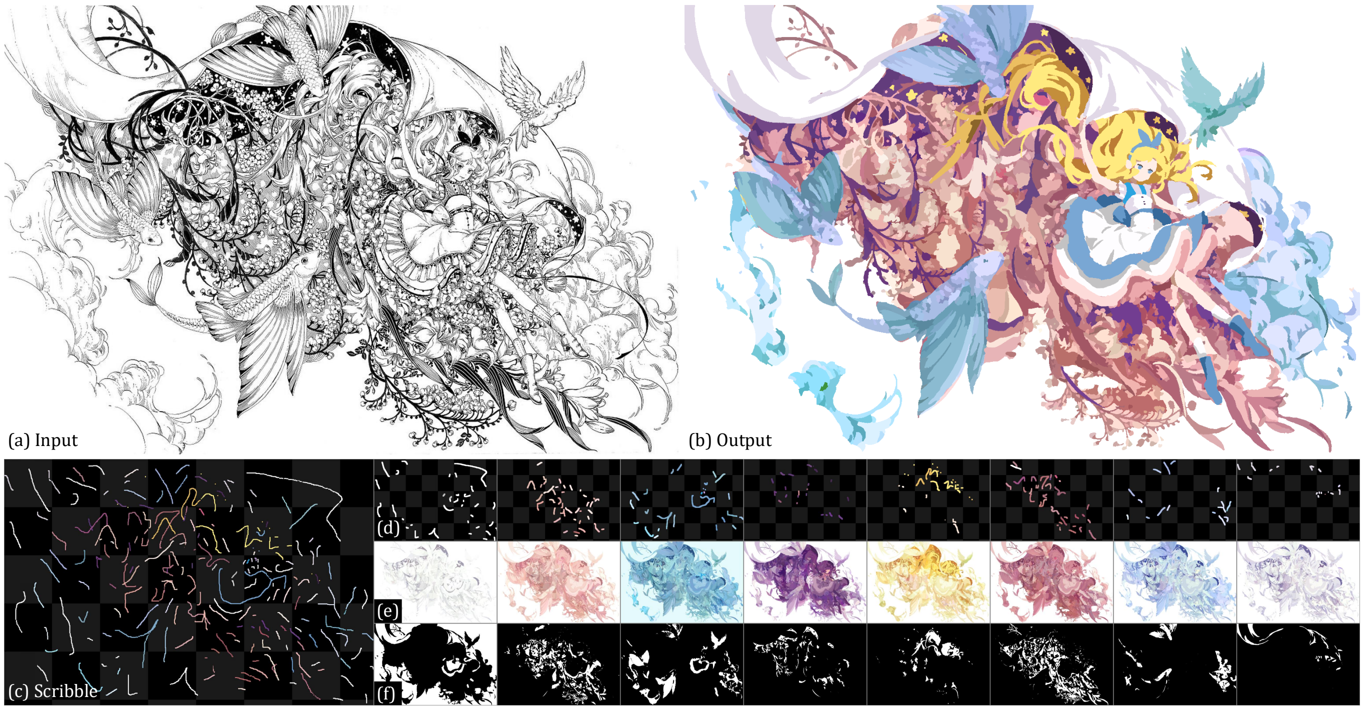


Fig. 9. **Amateur user result.** We show an example result achieved by an amateur user who first browses the auxiliary colour tables for inspiration, and then perform interactive colourisation of a complex line drawing in 8 minutes and 41 seconds. We show (a) the input line drawing, (b) output flat filling, (c) user scribbles, (d) split scribble maps $\boldsymbol{U}_i$, (e) colour maps $\boldsymbol{C}_i$, and (f) influence maps $\boldsymbol{I}_i$. *Sky with Alice © used with artist permission.*

also learning to perform a region segmentation and merging that improves the colourisation in a single framework.

4.5 Quantitative analysis

We also perform an analysis of the results obtained by the users by calculating how many regions are filled manually by user scribbles, or filled automatically by our framework. In particular, we divide all colourised regions into three categories:

(1) *Automatic regions.* The automatic regions are those regions with no colour indications, *i.e.*, regions that are not covered by any user scribbles. Our framework needs to automatically generate the colours for those regions.
(2) *Manual regions.* The manual regions are those regions coloured with accurate user-indicated colours. As mentioned in § 3.1.4, during the inference of our framework, we selectively merge some regions and replace their colours with the accurate scribble colours, according to Eq. (4) and the region structure estimated by the neural networks. This enables users to directly control and manipulate the precise colours of some specific regions, and those specific regions are viewed as manual regions.
(3) *Semi-automatic regions.* Excepting the automatic and manual regions, all the remaining regions are semi-automatic regions. The colours of those regions are governed by both the users and the neural networks. The users give rough scribbles with coarsely defined colours, and our framework generate visually satisfying colour transitions, variations, and gradients. Besides, as mentioned in § 3.1.7, the unreliable scribbles, *i.e.*, those scribbles with conflicting colours or boundary leakage, are also addressed smartly by our framework in this category.

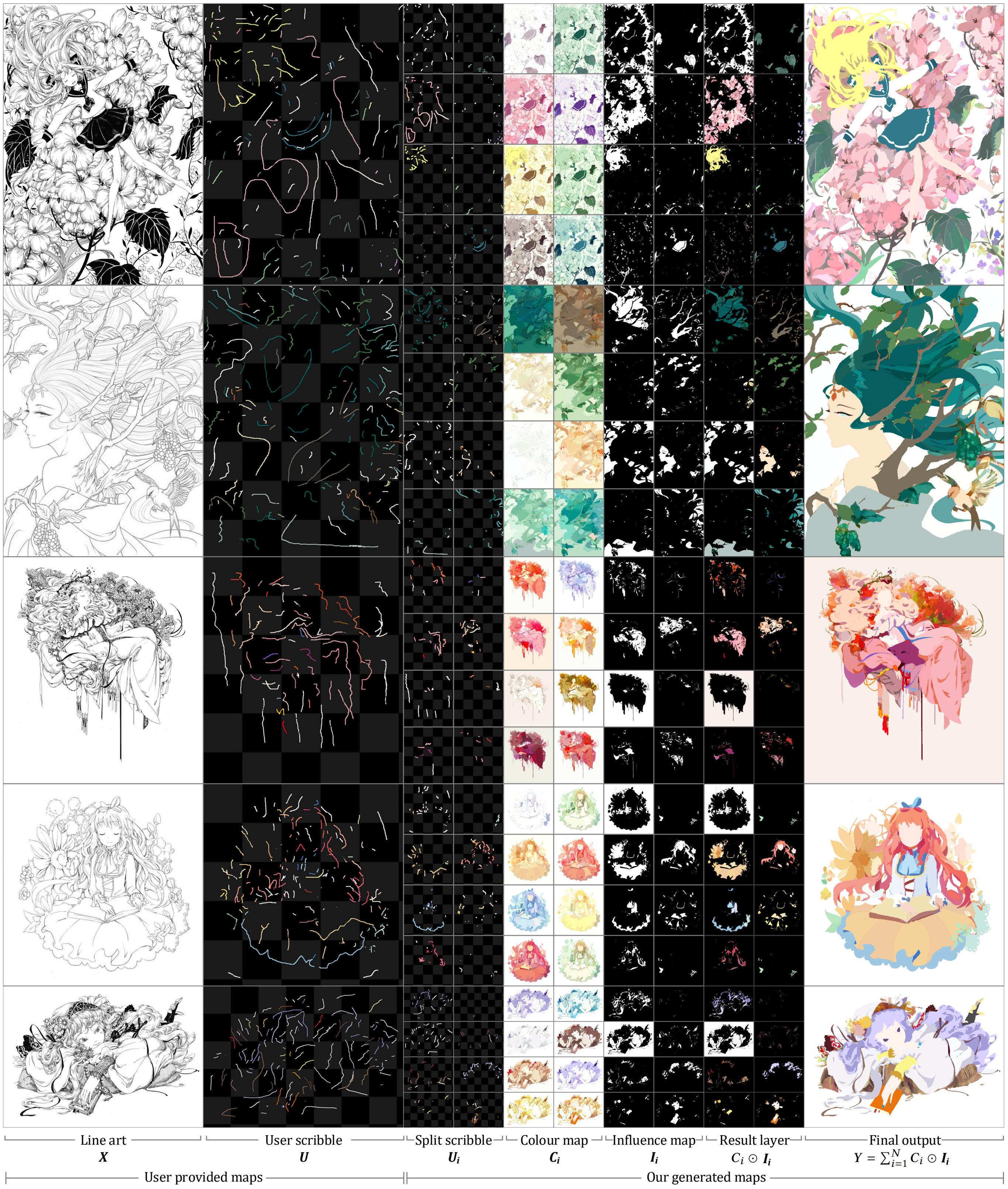


Fig. 10. **Qualitative results.** We show a break-down of several results with our proposed approach. More examples are provided in the supplementary material. *Flower with Alice, Tree Elves, Anna in Dream, Book Girl, and Reading Awake © used with artist permission.*

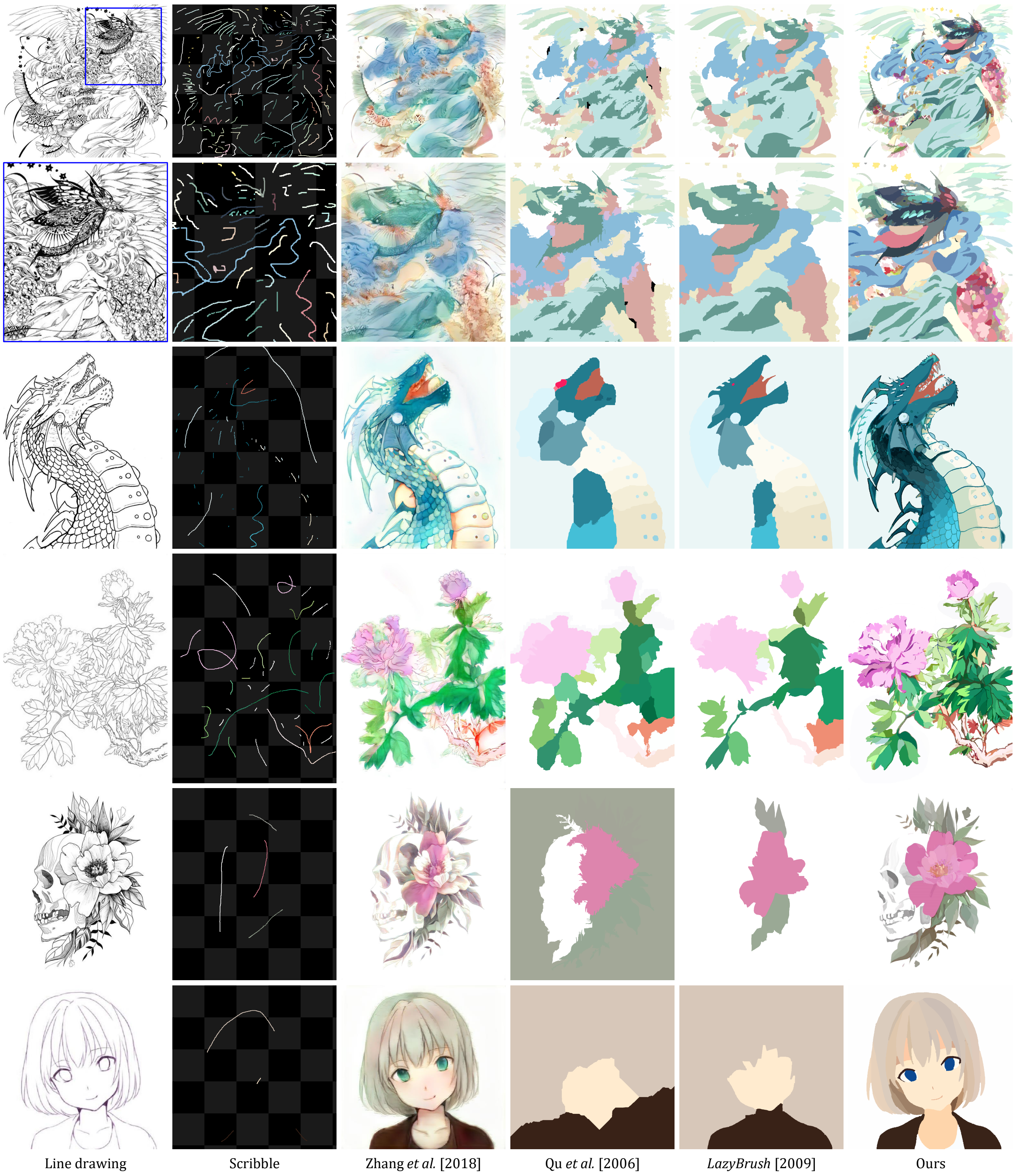


Fig. 11. **Comparison with existing colourisation approaches.** We compare our framework with [Qu et al. 2006; Sykora et al. 2009; Zhang et al. 2018]. *Flower Angel, Comollon, Wisteria Flowers, Poison Skull, and Megumi © used with artist permission.*

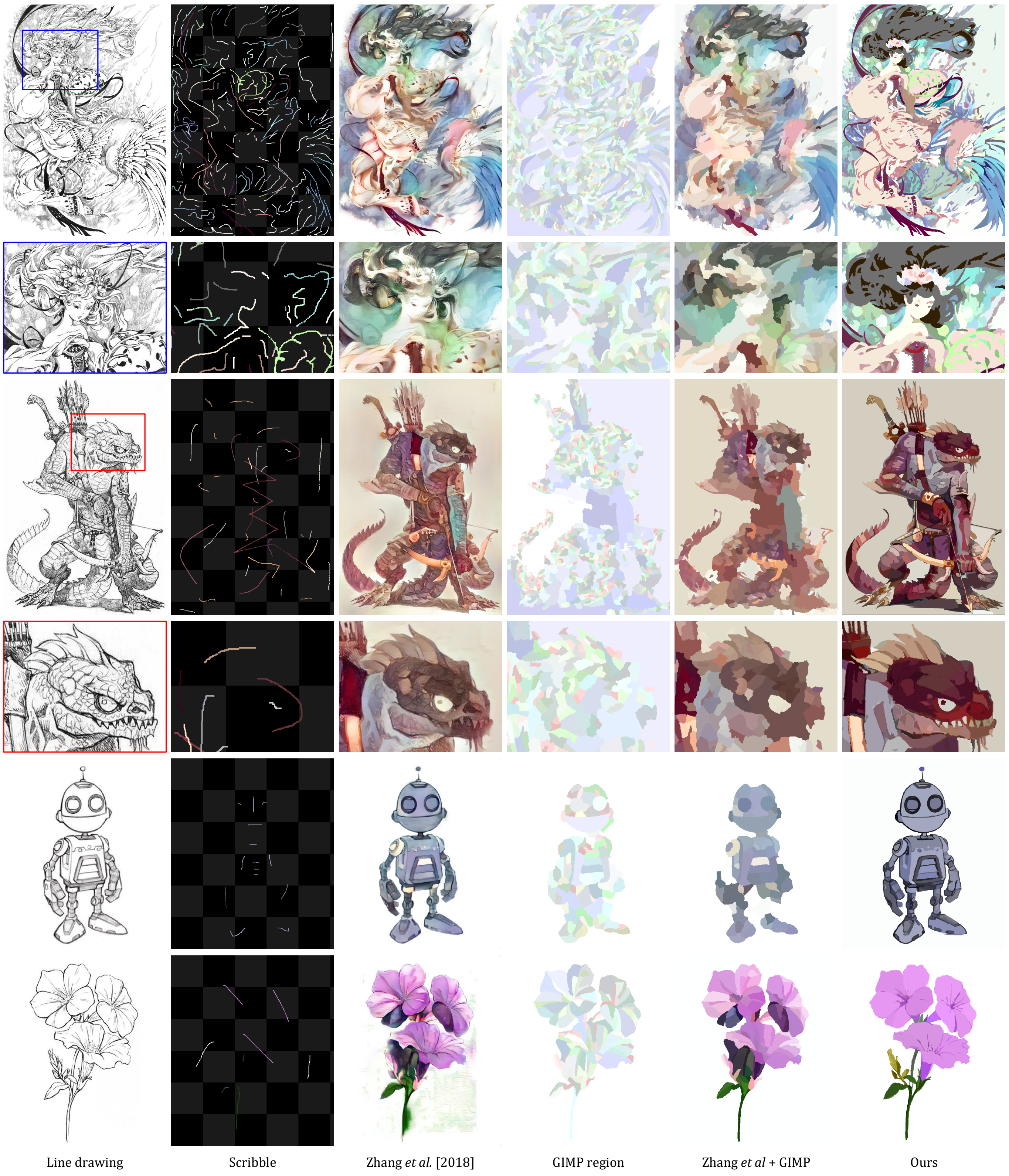


Fig. 12. **Comparison with combination of existing approaches for flat colourisation.** We compare our approach with the colourisation method [Zhang et al. 2018] combined with the segmentation method [Fourey et al. 2018]. *Fountain Angel, Goblin, Robot, and Azalea © used with artist permission.*

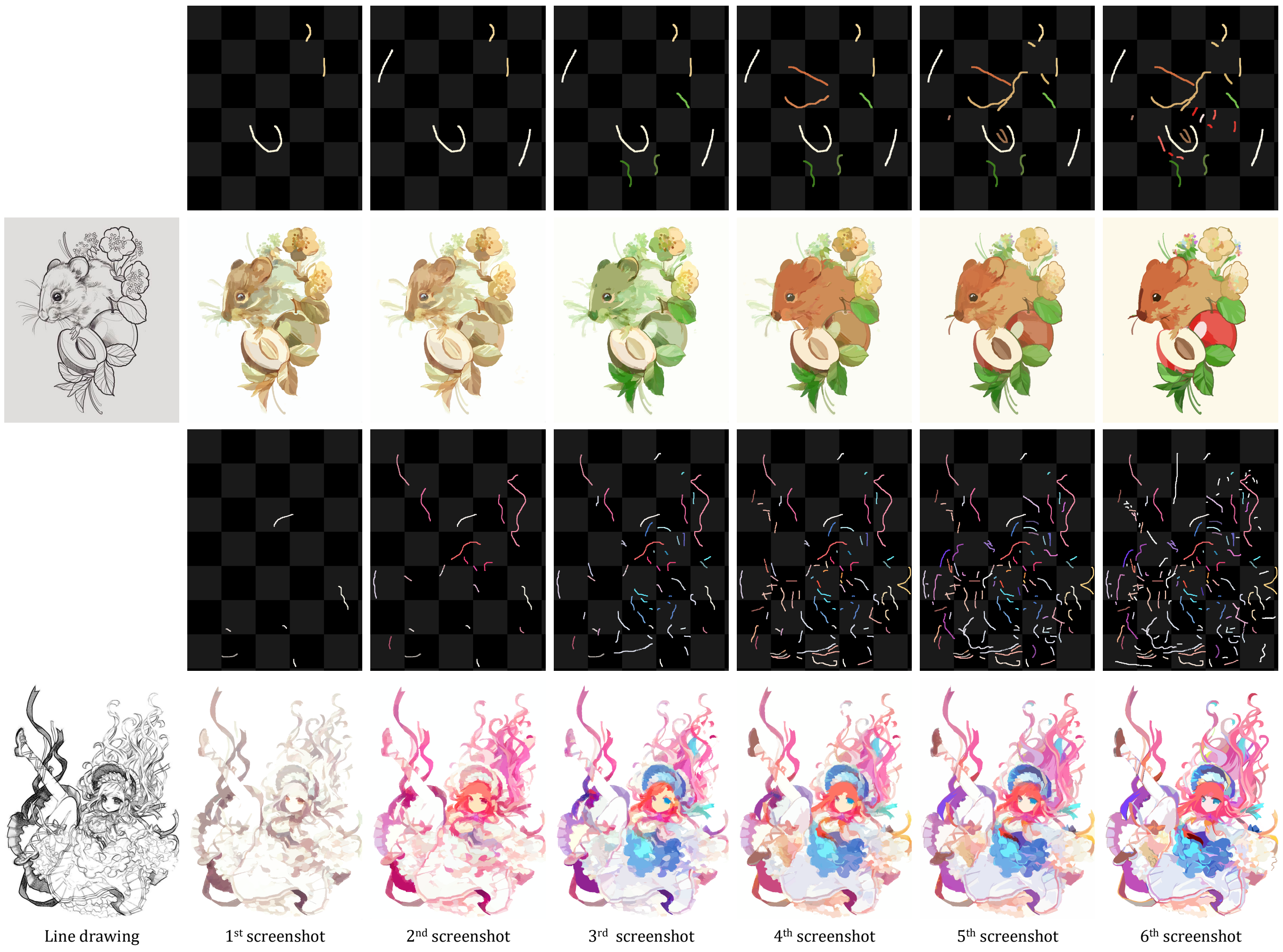


Fig. 13. **Snapshots during user interaction.** We present snapshots of the user scribble canvas and the result canvas captured during the interactive flat filling. *Squirrel, and Heavenly Girl © used with artist permission.*

Table 1. **Scribble analysis.** We perform an analysis of the number of scribbles used for the flat colourisation of different line drawings. We split the region colourisation into three categories: automatically filled regions, semi-automatically filled regions, and manually filled regions, depending on how the framework has colourised each region.

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% ± 5.98%	36.40% ± 5.44%	9.88% ± 3.30%

Results of this analysis are shown in Table 1, where we can see that most of the regions are automatically or semi-automatically coloured, with only roughly 10% of the regions being manually coloured. This highlights how satisfactory results can be obtained with a relatively small number of scribbles.

4.6 User interaction analysis

we sample several snapshots during the interactive flat filling and show them in Fig. 13. On one hand, we can see that users are able to progressively and interactively improve the flat filling results using our framework. On the other hand, our framework also does not fail even when the scribbles are minimal in the beginning. Furthermore, we also find that the users tend to draft up the initial colour compositions at the beginning time with a small number of scribbles, and then retouch the details with more scribbles.

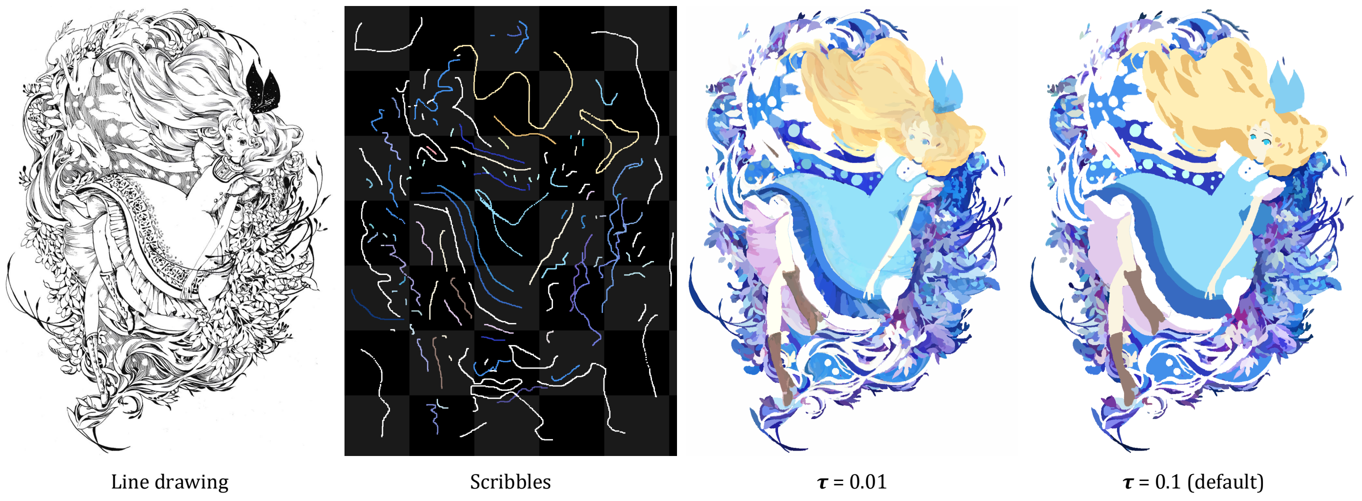


Fig. 14. **Analysis of the τ parameter.** The parameter τ controls how the framework merges adjacent regions. For higher values of τ, more regions will be merged. Different use cases may need different values of τ. *Alice's Night © used with artist permission.*

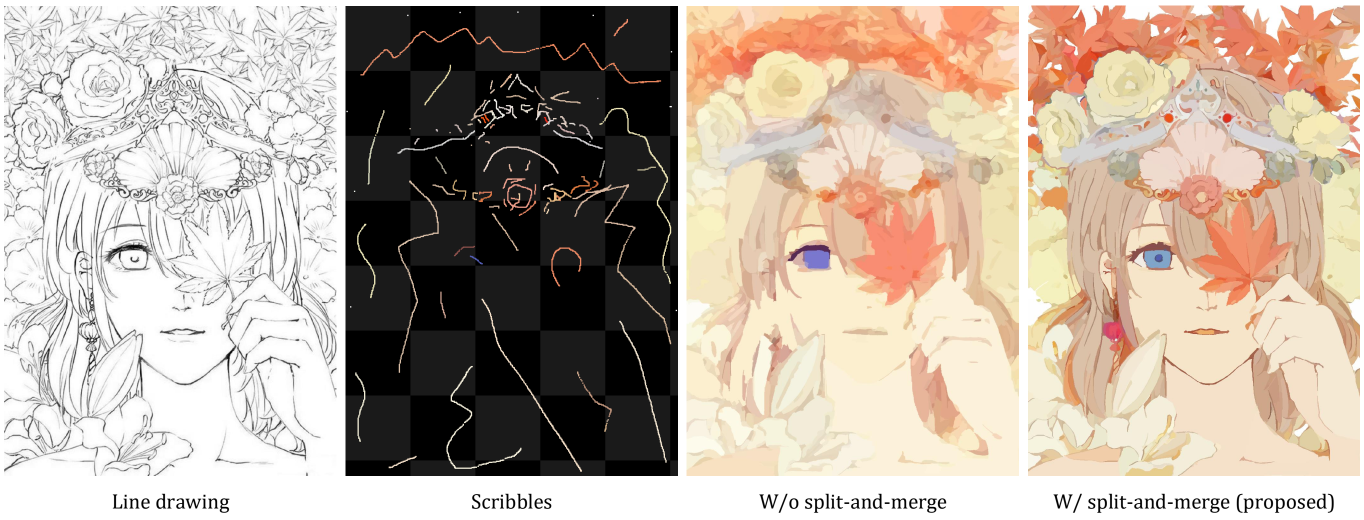


Fig. 15. **Significance of split filling mechanism.** We compare the results obtained from our neural architecture with (w/) or without (w/o) the split filling mechanism. *One Leaf Knows Autumn © used with artist permission.*

4.7 Ablative analysis

Our proposed approach not only depends on the user scribbles, but also has an important hyper-parameter τ that controls how adjacent regions are merged. As shown in Fig. 14, larger values of τ leads to more regions being merged, while low values can conserve too many details and lead to non-flat fillings. We find that $\tau = 0.1$ is a good compromise and use it as a default and recommended configuration.

4.8 Significance of split filling mechanism

We compare our framework with a cloned version but without the split-and-merge processing. To be specific, we train our neural architecture (Fig. 6) with non-split data to directly estimate the final colouring 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 colour map $\boldsymbol{C}'$. The influence decoder in Fig. 6 is unused and the influence map loss in Eq. (6) becomes zero. 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 colour consistency optimisation, remain same.

The results are shown in Fig. 15. We can see that the split-and-merge processing is an indispensable part of our framework. In absence of this processing, the outputs degenerate significantly yielding hardly useable flat filling. This is mainly because our split-and-merge processing mimics the real-life on-paper flat filling workflow with palette cells to manage colours and prevent unwanted

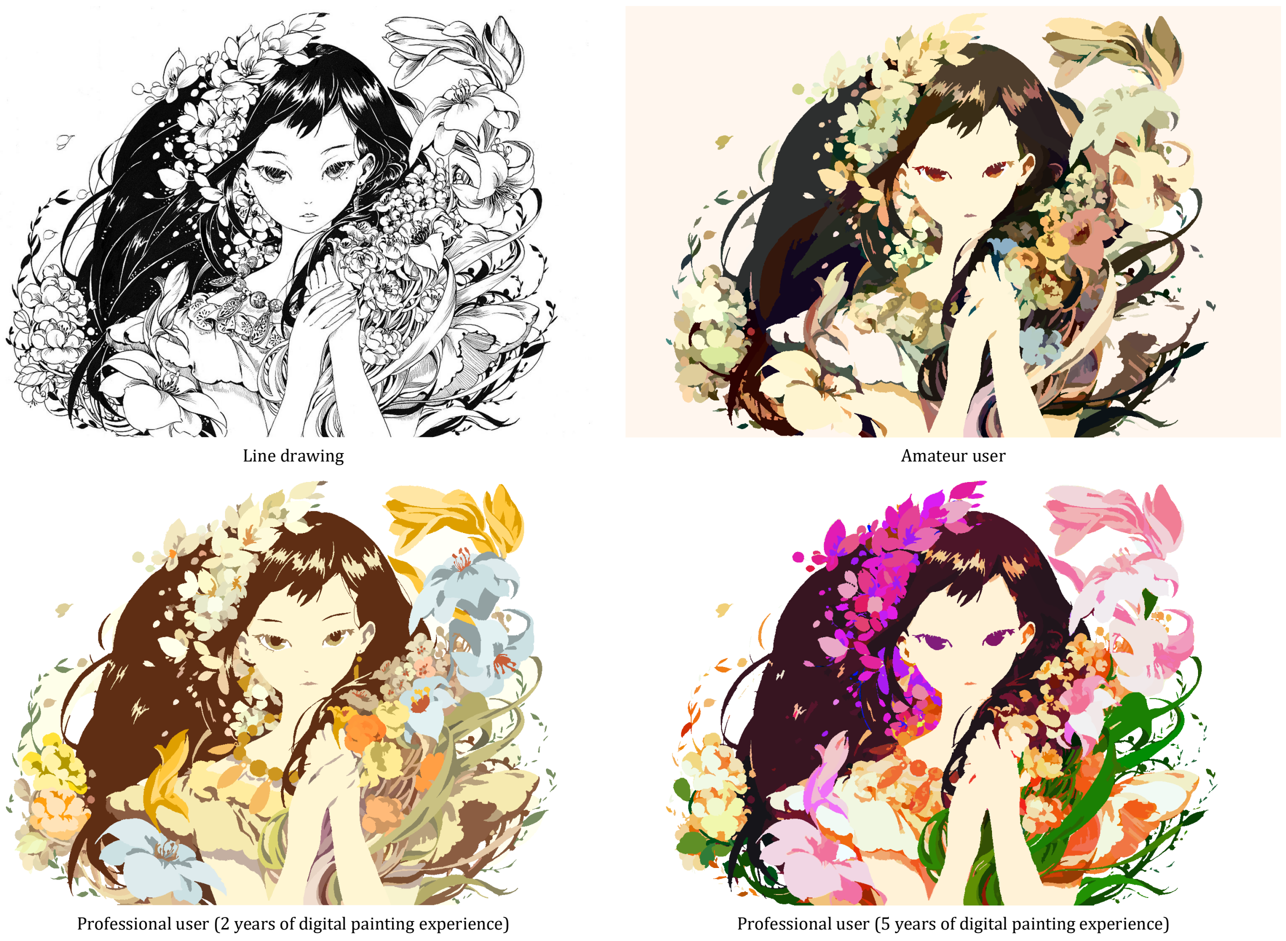

Line drawing
Amateur user
Professional user (2 years of digital painting experience)
Professional user (5 years of digital painting experience)

Fig. 16. **Comparison of results from amateur and professional users.** We show results created by one amateur user and two professional artists. All results are created using our framework. *Prayer © used with artist permission.*

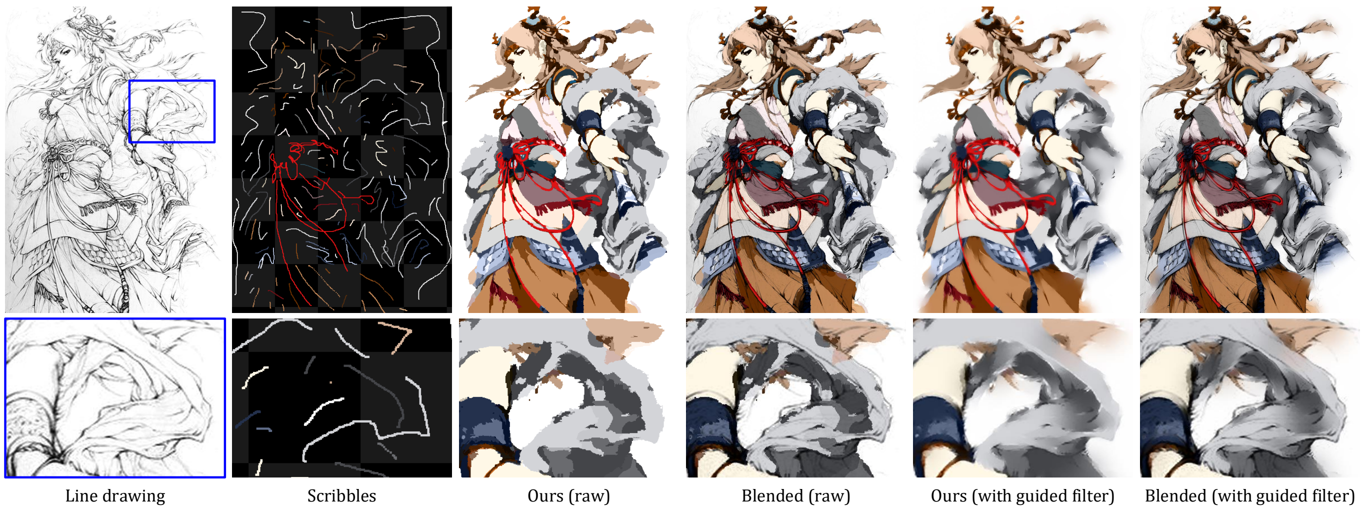


Fig. 17. **Limitation.** The generated regions can break the image structure, which is especially common when the input line drawing is a rough sketch. This limitation can be meliorated to some extent using guided filters [He et al. 2013]. *Zhao-Yun © used with artist permission.*

contamination between colour pigments. This characteristic is essential for professional flat filling studios.

4.9 Comparison of professional and amateur users

We show comparisons of results from professional and amateur users in Fig. 16, where all the results are achieved using our framework. We can see that the amateur user result is competitive to the professional ones, indicating that our framework has greatly helped the invited amateur user to perform flat filling. Also, we can see in this case that the result exhibit an accurate segmentation, and the generated colour variations are surprisingly effective.

4.10 Limitation

The major limitation of our framework is that, when the input line drawing is a rough sketch, the generated flat filling regions may not strictly fit the image structure as shown in Fig. 17. This causes aliasing distortion when the line drawing and the filled colours are blended. One possible solution for this limitation is to use guided filters [He et al. 2013] as a post-processing to improve the colour maps before blending but this can lead to other issues such as colour bleeding. How to further improve this structure consistency remains an open problem.

5 CONCLUSION

We present a line drawing flat filling approach motivated by the classic on-paper flat filling workflow. We observe the artist usage of the *grid palette*, and find that the colour pigments in the same palette cell can produce colour variations and transitions, whereas those in separate palette cells will not influence each other. Based on these observations, we propose to split the user scribble colours into a *virtual grid palette* to manage the colours. Afterwards, we perform independent colourisation in each palette cell and then merges them back together. Results show that our approach is able to handle diverse contents with complicated patterns and obtain reliable high-quality colourisations. We also show that the presented tool is helpful for both amateur users and professional artists.

ACKNOWLEDGMENTS

This technique is presented by Style2Paints Research. This technique belongs to Style2Paints Research Project Code (SRPC) HaPa — "Project HaPa: Happy Painting!" We thank all anonymous reviewers and chair members for evaluating and improving this work. We thank all artists who have help us in achieving this work. This work is also supported by JST PRESTO (Simo-Serra, Grant Number: JP-MJPR1756).

REFERENCES

Yagiz Aksoy, Tung Ozan Aydin, Aljosa Smolic, and Marc Pollefeys. 2017. Unmixingbased soft color segmentation for image manipulation. *ACM Transactions on Graphics* (2017).

J.T. Barron and J. Malik. 2012. Color constancy and intrinsic images and shape estimation. *ECCV* (2012).

H. G. Barrow and J. M. Tenenbaum. 1978. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, A. Hanson and E. Riseman (Eds.). Academic Press, 3–26.

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

DanbooruCommunity. 2018. Danbooru2017: A Large-Scale Crowdsourced and Tagged Anime Illustration Dataset.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. 2004. Efficient Graph-Based Image Segmentation. *IJCV* (2004).

SÃĪbastien Fourey, David Tschumperle, and David Revoy. 2018. A Fast and Efficient Semi-guided Algorithm for Flat Coloring Line-arts. *EUROGRAPHICS* (2018).

Kevin Frans. 2017. Outline Colorization through Tandem Adversarial Networks. *In Arxiv* (2017).

Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. 2017. Comicolorization: semi-automatic manga colorization. In *SIGGRAPH Asia 2017 Technical Briefs*.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *CVPR*. 2414–2423.

P.V. Gehler, C. Rother, M. Kiefel, L. Zhang, and B. Scholkopf. 2011. Recovering intrinsic images with a global sparsity prior on reflectance. *NIPS* (2011).

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *NIPS* 3 (2014), 2672–2680.

Kaiming He, Jian Sun, and Xiaoou Tang. 2013. Guided Image Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 6 (jun 2013), 1397–1409.

Mingming He, Dongdong Chen, Jing Liao, Pedro V Sander, and Lu Yuan. 2018. Deep Exemplar-based Colorization. *ACM Transactions on Graphics* (2018).

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

Paulina Hensman and Kiyoharu Aizawa. 2017. cGAN-based Manga Colorization Using a Single Training Image. *In Arxiv* (2017).

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. 2018. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *ICML*.

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2016. 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).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *Computer Science* (2014).

Yuki Koyama and Masataka Goto. 2018. Decomposing Images into Layers with Advanced Color Blending. *Computer Graphics Forum* 37, 7 (2018), 397–407.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Learning Representations for Automatic Colorization. In *ECCV*. Springer, 577–593.

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

Chengze Li, Xueting Liu, and Tien-Tsin Wong. 2017. Deep Extraction of Manga Structural Lines. *ACM Transactions on Graphics* 36, 4 (2017).

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

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. 2017. Focal loss for dense object detection. *ICCV* (2017).

Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. 2015. Closure-aware Sketch Simplification. *ACM Transactions on Graphics* 34, 6 (November 2015), 168:1–168:10.

Yifan Liu, Zengchang Qin, Zhenbo Luo, and Hua Wang. 2017. Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks. *In Arxiv* (2017).

I. Omer and M. Werman. 2004. 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. https://doi.org/10.1109/cvpr.2004.1315267

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

Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. 2017. Scribbler: Controlling Deep Image Synthesis with Sketch and Color. *CVPR* (2017).

M. Serra, O. Penacchio, R. Benavente, and M. Vanrell. 2012. Names and shades of color for intrinsic image estimation. *CVPR* (2012).

John R. Shaw. 2004. QuickFill: An Efficient Flood Fill Algorithm. *codeproject* (2004).

Jianbing Shen, Xiaoshan Yang, Yunde Jia, and Xuelong Li. 2011. Intrinsic Images Using Optimization. *CVPR* (2011).

Min Shi, Jia-Qi Zhang, Shu-Yu Chen, Lin Gao, Yu-Kun Lai, and Fang-Lue Zhang. 2020. Deep Line Art Video Colorization with a Few References. *Arxiv* (2020).

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

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018b. Real-Time Data-Driven Interactive Rough Sketch Inking. *ACM Transactions on Graphics* (2018).

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup. *ACM Transactions on Graphics* 35, 4 (2016).

Daniel Sykora, John Dingliana, and Steven Collins. 2009. LazyBrush: Flexible Painting Tool for Hand-drawn Cartoons. *Computer Graphics Forum* 28, 2 (2009).

TaiZan. 2016. PaintsChainer Tanpopo. *PreferredNetwork* (2016).

Stephen Gary Wozniak. 1976. The Apple I Computer. *Apple* (1976).
Lvmin Zhang, Yi JI, and Chunping Liu. 2020. DanbooRegion: An Illustration Region Dataset, In European Conference on Computer Vision (ECCV). *ECCV*.
Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. 2018. Two-stage sketch colorization. In *ACM Transactions on Graphics*.
Qing Zhang, Chunxia Xiao, Hanqiu Sun, and Feng Tang. 2017a. Palette-Based Image Recoloring Using Color Decomposition Optimization. *IEEE Transactions on Image Processing* (2017).
Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful Image Colorization. In *ECCV*.
Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. 2017b. Real-Time User-Guided Image Colorization with Learned Deep Priors. *ACM Transactions on Graphics* 9, 4 (2017).
T. Y. Zhang and C. Y. Suen. 1984. A fast parallel algorithm for thinning digital patterns. *Commun. ACM* (1984).
Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. 2017. Toward multimodal image-to-image translation. In *NIPS*.