Generating Manga from Illustrations via Mimicking Manga Creation Workflow

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

Xinrui Wang
The University of Tokyo / Tencent
China
secret_wang@outlook.com

Qingnan Fan
Stanford University
United States
fqnchina@gmail.com

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

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

Abstract

We present a framework to generate manga from digital illustrations. In professional mange studios, the manga create workflow consists of three key steps: (1) Artists use line drawings to delineate the structural outlines in manga storyboards. (2) Artists apply several types of regular screentones to render the shading, occlusion, and object materials. (3) Artists selectively paste irregular screen textures onto the canvas to achieve various background layouts or special effects. Motivated by this workflow, we propose a data-driven framework to convert a digital illustration into three corresponding components: manga line drawing, regular screentone, and irregular screen texture. These components can be directly composed into manga images and can be further retouched for more plentiful manga creations. To this end, we create a large-scale dataset with these three components annotated by artists in a human-in-the-loop manner. We conduct both perceptual user study and qualitative evaluation of the generated manga, and observe that our generated image layers for these three components are practically usable in the daily works of manga artists. We provide 60 qualitative results and 15 additional comparisons in the supplementary material. We will make our presented manga dataset publicly available to assist related applications.

1. Introduction

Generating manga from illustrations (Fig. 1-left) is an important task in high demand. The expansion of manga market and the rarity of manga artist have caused many manga companies to recruit a large number of digital painting or illustration artists and train them as manga creators. Training an artist to master the unique manga workflow, *e.g.*, inking strategy, screentone management, texture applying, *etc.*, is financially expensive and can often take weeks or even months. To reduce such training costs and speed up the producing, many manga companies have began to adopt techniques that generate manga from generic art forms like illustrations and digital paintings, so that the costs to train artists can be saved, and those newly hired digital illustration artists can be free from learning extra skills and can create manga directly in their familiar digital illustration working environments. The software Clip Studio Paints (CSP) [6] and Adobe After Effects (AE) [1] are typical examples with many plugins and online tutorials [10, 11, 7, 9, 8] for editing illustrations to obtain manga manually. The widespread popularity of those tutorials and plugins verifies the significance of the problem to generate manga from illustrations.

Figure 1. Our framework automatically generate the right high-quality manga given the left illustration. Zoom in to see details.

This paper starts with a key observation: the unique visual appearance of manga comes from the unique manga creation workflow. As shown in Fig. 2, we verify this observation

(a) Inking line drawing (b) Pasting regular screentone (c) Pasting irregular screen texture

Figure 2. Justification for our motivation: the classic professional manga creation workflow. Artists first ink structural line drawings, and then apply regular screentones and irregular screen texture to obtain the final manga.

by studying the manga creation workflow in professional studios. Firstly, artists draw line drawings as the initial outlines in manga storyboards, *e.g.*, the artist delineates the line structure of the girl portrait in Fig. 2-(a). Secondly, artists paste screentone sheets with different regular patterns onto the regions between lines, *e.g.*, the hair, eyes and dress in Fig. 2-(b) are pasted with such screentone sheets. Thirdly, artists fill the canvas with irregular screen textures to achieve background layouts or special effects, *e.g.*, the artist applies the romantic background screen texture to set off the mood of the girl character in Fig. 2-(c). We can see that these three steps are sufficient and necessary to determine the appearance of a manga image, and each step is indispensable for preparing the high-quality manga product.

Might we be able to achieve a program that can mimic the above workflow, producing manga images with similar appearance to the ones created by artists manually with this workflow, and at the same time, yielding independently usable result layers that can assist artists in each step of this workflow? To achieve these goals, we present a deep learning approach to mimic the manga creation workflow step-by-step. Firstly, given an input illustration, our framework estimates a line drawing map that plays a similar role to the line drawings inked by artists manually. Secondly, our framework segments the input image into a fixed number of screentone classes, and pastes corresponding regular screentone sheets onto the image regions of each class. Thirdly, our framework predicts a texture mask to identify the areas that need to be pasted with screen textures, and afterwards synthesizes irregular textures in the identified areas. In this way, our framework automatically produces the line drawings, regular screentones, and screen textures. Those components can be independently used by artists for further creation, or can be directly composed into the manga outputs.

To this end, we invite artists to annotate a large-scale dataset and learn a hierarchical neural network in a data-driven manner. Our dataset contains 1502 image&annotation pairs of {illustration image, line drawing annotation, regular screentone segmentation annotation, and irregular screen texture mask annotation}. All annotations are achieved with a human-in-the-loop approach, and checked by multiple artists for quality assurance. We will make this dataset publicly available to assist related applications.

Experiments show that mimicking the manga creation workflow yields several advantages. Firstly, in qualitative analyse, our framework can produce not only single manga image but also independent image layers at each workflow step to assist artists. Then, in perceptual user study, our framework tends to learn the artist decisions recorded in our presented dataset for each workflow step, making our framework preferred by the artists, as the manga creation depends heavily on content semantics and even artist perception. Furthermore, we provide 60 qualitative results and 15 additional comparisons in the supplementary material.

In summary, our contributions are: (1) We propose a data-driven framework to generate manga from illustrations by mimicking the professional manga creation workflow, including the steps of line drawing inking, regular screentone pasting, and irregular screen texture pasting. (2) We present a large-scale artistic dataset of illustration and annotation pairs to facilitate the problem to generate manga from illustrations and assist related applications. (3) Perceptual user study and qualitative evaluations demonstrate that our framework is more preferable by artists when compared to other possible alternatives.

2. Related Work

Screentone and manga processing. The synthesis of manga screentone or halftoned texture is a unique problem with considerable demand. Halftoning exploits the spatial integration of human vision to approximate the intensity over a small local region with black-and-white pixels [24, 16, 44, 21]. Path-based methods [45, 32] try to reduce the artifact patterns by adjusting the scanning path over images. Knuth *et al*. [24] enhance edges in a prepro-

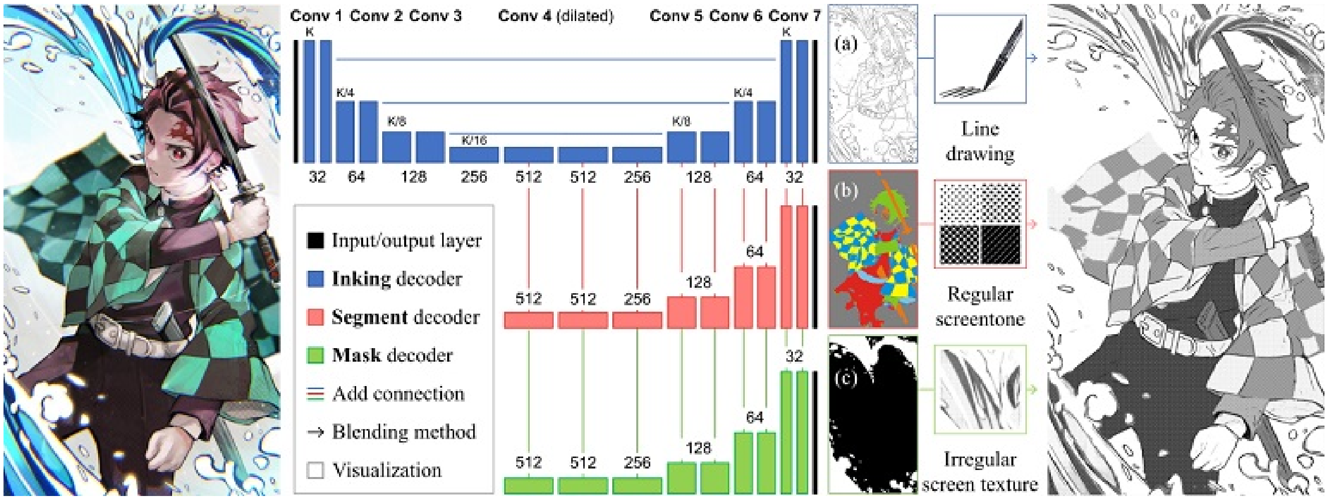


Figure 3. Overview of our framework. Given an input illustration, our framework separately estimates the line drawing, regular screentone segmentation, and the irregular screen texture mask. These components are composed to produce the final manga result. All convolutional layers use 3×3px kernels and are processed by Batch Normalizations (BNs) and ReLU activations.

cessing step to preserve the edges. Afterwards, Buchanan *et al*. [3] preserve the fine structure by optimising the structure similarity. Perception-aware methods [51, 52, 34, 36] map pixel-wise or region-wise luminance to screentone patterns to achieve perceptually distinguishable salient screentone structure. Learning-based method Li *et al*. [26] train neural networks to predict the sketching texture in an end-to-end manner. Variational-auto-encoder screentone filler [53] maps the screened manga to an intermediate domain. Hatching is another technique to halftoned effects. Winkenbach *et al*. [50] synthesise pen-and-ink illustrations by rendering a geometric scene with prioritised stroke textures. Our framework not only focuses on the synthesis of screentones, but also works with the practical workflow of manga creation, including the steps of line drawing inking, screentone synthesis, and texture blending.

Cartoon and digital painting techniques. Cartoon image processing and computational digital painting have been extensively studied in the past few decades. Manga structure extraction [25], cartoon inking [40, 38, 39], and line closure [29, 31] methods analysis the lines in cartoon and digital paintings. A region-based composition method can be used in cartoon image animating [41]. Stylization methods [5, 48, 54, 56, 55] generate cartoon images or artistic drawings from photographs or human portraits. Line drawing colour filling applications [58, 43, 42] colourize sketch or line drawings with optimization-based or learning-based approaches. Our approach generates manga from illustrations and digital paintings, and can be used in manga products and related cartoon or digital painting applications.

Image-to-image translation and stylization. The task of generating manga from illustrations can also be seen as an image-to-image translation or image stylization problem. For example, paired image-to-image translation [20, 47, 4] and unpaired methods [59, 22, 2, 57, 14, 30]. These methods can transform images across categories, *e.g.*, *maps-to-aerials*, *edges-to-cats*, and in our case, *illustrations-to-manga*. Neural style transfer [17, 19, 27] can stylize images via transferring low-level style features to the content images. Our experiments include typical candidates of these methods, analysing the differences between our customized framework and these generic methods.

3. Method

We present an overview of our framework as shown in Fig. 3, where the input is an illustration $\boldsymbol{X} \in \mathbb{R}^{w \times h \times c}$ (Fig. 3-left), whereas the output is a manga image $\boldsymbol{Y} \in \mathbb{R}^{w \times h}$ (Fig. 3-right), with w, h, c being the image width, height, and channel quantity. We train Convolutional Neural Networks (CNNs) to mimic the professional manga creation workflow with the following three steps:

Firstly, to mimic the behaviour that artists ink line drawings to delineate the structural outlines in manga storyboards (Fig. 2-(a)), our framework estimates a line drawing map $\hat{\boldsymbol{L}} \in \mathbb{R}^{w \times h}$ (Fig. 3-(a)). Learnt from thousands of images, our neural network inks salient structural lines and suppresses unwanted constituents like noises and shadow interferences.

Secondly, to mimic the behaviour that artists paste regular screentones to depict the manga shading and object materials (Fig. 2-(b)), our framework estimates a panoptic segmentation map, namely the regular screentone segmentation map $\hat{\boldsymbol{S}} \in \mathbb{R}^{w \times h \times N}$ (Fig. 3-(b)), with N representing the number of screentone classes. Such segmentation yields a set of region labels, and each label indicates one type of screentone that should be pasted to each region. Our framework then pastes the screentones according to such region-wise labels.

Thirdly, to mimic the behaviour that artists paste irregular screen textures to achieve background layouts or special effects (Fig. 2-(c)), our framework estimates an irregular texture mask $\hat{\boldsymbol{M}} \in \mathbb{R}^{w\times h}$ (Fig. 3-(c)) to identify the areas that should be covered with irregular textures, and afterwards synthesizes manga textures for those identified areas.

Finally, the output manga image $\boldsymbol{Y}$ can be composed with

$$\boldsymbol{Y} = \hat{\boldsymbol{L}} \odot \varphi_{\text{tone}}(\hat{\boldsymbol{S}}) \odot \varphi_{\text{tex}}(\hat{\boldsymbol{M}}, \boldsymbol{X})\,, \tag{1}$$

where $\odot$ is Hadamard product, $\varphi_{\text{tone}}(\cdot)$ is a tone transform (described in § 3.2) that pastes screentone sheets according to the given screentone segmentation $\hat{\boldsymbol{S}}$, and $\varphi_{\text{tex}}(\cdot,\cdot)$ is a texture transform (described in § 3.3) that synthesizes textures given the texture mask $\hat{\boldsymbol{M}}$ and the illustration $\boldsymbol{X}$.

In order to train this framework, we invite artists to annotate a dataset containing 1502 pairs of {illustration $\boldsymbol{X}$, line drawing map $\boldsymbol{L}$, regular screentone segmentation map $\boldsymbol{S}$, and irregular texture mask $\boldsymbol{M}$}. These data are annotated in a human-in-the-loop manner, *i.e.*, artists create annotations for our framework to learn, and our framework estimates coarse annotations for artists to refine. The annotating approach is detailed in § 4.

3.1. Inking line drawing

When creating manga in real life, artists ink line drawings to outline the object structures in manga storyboards. Our framework estimates a line drawing map $\hat{\boldsymbol{L}}$ to mimic this artist behaviour. Given the ground truth line drawing $\boldsymbol{L}$ and the estimation $\hat{\boldsymbol{L}}$, we customize a likelihood $\mathcal{L}_{\text{ink}}$ with

$$\mathcal{L}_{\text{ink}} = \sum_p \Big(||\hat{\boldsymbol{L}}_p - \boldsymbol{L}_p||_2^2 + \lambda_i ||\phi(\hat{\boldsymbol{L}})_p - \phi(\boldsymbol{L})_p||_2^2 \Big)\,, \tag{2}$$

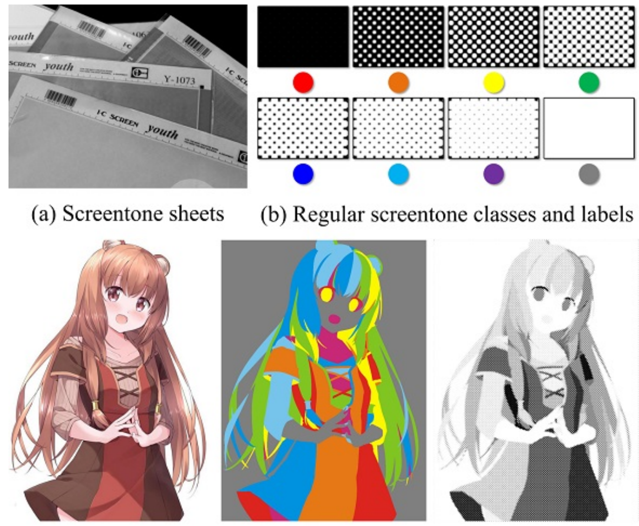
where p is pixel position, $||\cdot||_2$ is Euclidean distance, and λ_i is a weighting parameter. The operator $\phi(\cdot)$ is a high-pass transform that penalizes the line patterns — we observe that the line patterns in line drawings routinely come with sparse and discrete high-amplitude/frequency transitions over pixel intensities, and we tailor-make the transform $\phi(\cdot)$ to identify such line patterns that are "darker" than their surrounding low-frequency domain with

$$\phi(\hat{\boldsymbol{L}})_p = \begin{cases} ||\hat{\boldsymbol{L}}_p - g(\hat{\boldsymbol{L}})_p||_2, & \text{if} \quad \hat{\boldsymbol{L}}_p < g(\hat{\boldsymbol{L}})_p; \\ 0, & \text{others}, \end{cases} \tag{3}$$

where $g(\cdot)$ is a Gaussian filter with a sigma of 3.0. With this transform, the likelihood $\mathcal{L}_{\text{ink}}$ not only describes how close the estimation $\hat{\boldsymbol{L}}$ is to the ground truth $\boldsymbol{L}$, but also penalizes the lines guided by the transform $\phi(\cdot)$.

3.2. Pasting regular screentone

Manga artists paste screentone sheets with regular patterns onto their canvases to render the shading, occlusion, and object materials. A widely-used commercial screentone standard "JAPAN-DELETER-SE" [13] (Fig. 4-(a)) includes 8 types of manga screentone sheets (Fig. 4-(b)). Based on this standard, our framework estimates a screentone segmentation map $\hat{\boldsymbol{S}}$ (Fig. 4-(d)) with 8 classes corresponding to these 8 types of screentones. With the estimated logits $\hat{\boldsymbol{S}}$ and the ground truth label $\boldsymbol{S}$, we measure the likelihood $\mathcal{L}_{\text{seg}}$ with the Softmax Cross Entropy [49] as



(a) Screentone sheets (b) Regular screentone classes and labels

(c) Illustration (d) Region-wise label (e) Pasted screentone

Figure 4. Pasting regular screentone. (a) Examples of real-life screentone sheets. (b) Standard manga screentone classes and corresponding label colours. (c) Example illustration. (d) Screentone segmentation map annotated by artists manually. (e) Screentones pasted according to the segmentation.

$$\mathcal{L}_{\text{seg}} = -\sum_{p,i} \log(\psi(\hat{\boldsymbol{S}}_p) \odot \boldsymbol{S}_p)_i\,, \tag{4}$$

where $\psi(\cdot)$ is Softmax [49] operation, p is pixel position, and i is class channel index. We further observe how artists paste screentone sheets, and find that artists are accustomed to paste screentones in a region-wise manner instead of in pixel-wise, *i.e.*, artists paste screentone sheets region-by-region rather than pasting independent sheets for each individual pixel. To achieve such region-wise consistency, we use the Felzenszwalb [15] super-pixel algorithm to segment the input image $\boldsymbol{X}$ into a set of over-segmented regions $\boldsymbol{\Omega} = \{\omega_{1\ldots n}\}$, and penalize the region-wise variation $\mathcal{L}_{\text{var}}$ with

$$\mathcal{L}_{\text{var}} = \sum_p ||\hat{\boldsymbol{S}}_p - \overline{\hat{\boldsymbol{S}}_{\omega(p)}}||_2^2\,, \tag{5}$$

where $\omega(p)$ is a super-pixel region in $\boldsymbol{\Omega}$ that the pixel p belongs to, and $\overline{\hat{\boldsymbol{S}}_{\omega(p)}}$ is the average value of $\hat{\boldsymbol{S}}$ in the region $\omega(p)$. By encouraging the region-wise consistency, this penalty mimics the artist behaviour of region-by-region

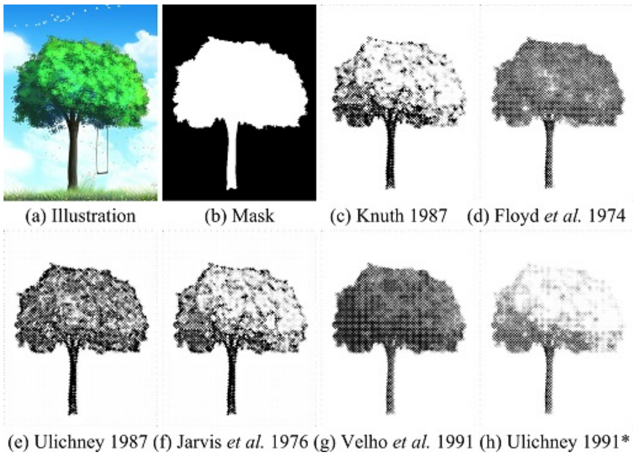


Figure 5. Pasting irregular screen texture. (a) Example illustration. (b) Screen texture mask annotated by artist manually. (c-h) Different types of halftone transforms. The "*" indicates the default transform used by our framework.

screentone sheet pasting. Afterwards, our framework pastes the screentones according to the segmentation $\hat{\boldsymbol{S}}$ with a tone transform $\varphi_{\text{tone}}(\cdot)$ as

$$\varphi_{\text{tone}}(\hat{\boldsymbol{S}})_p = \sum_i \psi(\hat{\boldsymbol{S}}_p)_i \odot (\boldsymbol{T}_i)_p \,, \tag{6}$$

where $\boldsymbol{T}_i \in \mathbb{R}^{w \times h}$ is a screentone image computed by tiling the i-th screentone patch in Fig. 4-(b). We show an example of the transform $\varphi_{\text{tone}}(\cdot)$ in Fig. 4-(e), where the screentone segmentation (Fig. 4-(d)) is transformed using the screentone-label correspondence in Fig. 4-(b).

3.3. Pasting irregular screen texture

The screen texture pasting is an indispensable step in the manga creation workflow. Such screen textures have irregular patterns and can be used in scenarios like background layouts or special effects. Our framework identifies the areas that need to be pasted with screen texture by estimating a texture mask $\hat{\boldsymbol{M}}$ (Fig. 5-(b)). We minimize the likelihood $\mathcal{L}_{\text{mask}}$ between the estimation $\hat{\boldsymbol{M}}$ and the ground truth $\boldsymbol{M}$ with

$$\mathcal{L}_{\text{mask}} = \sum_p ||\hat{\boldsymbol{M}}_p - \boldsymbol{M}_p||_2^2 \,, \tag{7}$$

where p is pixel position. Furthermore, we observe how artists paste screen textures, and find that artists tend to paste identical texture in consecutive and monotonic areas, *e.g.*, large backgrounds, full-screen effects, big fonts, *etc*. To achieve the spacial coherency in such areas, we introduce an anisotropic penalty $\mathcal{L}_{\text{ani}}$ with

$$\mathcal{L}_{\text{ani}} = \sum_p \sum_{i \in o(p)} \sum_{j \in o(p)} \left(\delta(\boldsymbol{X})_{ij} ||\hat{\boldsymbol{M}}_i - \hat{\boldsymbol{M}}_j||_2^2 \right) \,, \tag{8}$$

where $o(p)$ is a 3×3 window centred at the pixel position p, with $\delta(\cdot)$ being an anisotropic term $\delta(\boldsymbol{X})_{ij} = \exp(-||\boldsymbol{X}_i - \boldsymbol{X}_j||_2^2/\kappa^2)$, and κ is an anisotropic weight. The term $\delta(\boldsymbol{X})_{ij}$ increases when $o(p)$ is located at consecutive and monotonic areas with uniform pixel intensities, and decreases when $o(p)$ comes across salient contours like edges. Afterwards, given the estimated mask $\hat{\boldsymbol{M}}$, our framework synthesizes manga textures in the masked areas with a texture transform $\varphi_{\text{tex}}(\cdot, \cdot)$ as

$$\varphi_{\text{tex}}(\hat{\boldsymbol{M}}, \boldsymbol{X})_p = \sum_p \xi(\boldsymbol{X})_p \odot \hat{\boldsymbol{M}}_p \,, \tag{9}$$

where $\xi(\cdot)$ is a halftone transform to synthesize texture. Our framework allows the transform $\xi(\cdot)$ to be chosen from many popular halftone synthesizing algorithms [24, 16, 44, 21, 45, 32] as shown in Fig. 5-(c-h), and we use the dotted transform [44] (Fig. 5-(h)) by default.

3.4. Neural architecture and learning objective

Our neural network architecture consists of three convolutional decoders: inking decoder, segment decoder, and mask decoder (Fig. 3). We apply Batch Normalization [18] and ReLU activation [33] to each convolutional layer. The sampling layers have skip connections in U-net [37] style. The three decoders are trained jointly with the loss term $\mathcal{L}$ with

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{ink}} + \lambda_2 \mathcal{L}_{\text{seg}} + \lambda_3 \mathcal{L}_{\text{mask}} + \lambda_{\text{var}} \mathcal{L}_{\text{var}} + \lambda_{\text{ani}} \mathcal{L}_{\text{ani}} \,, \tag{10}$$

where $\lambda_{1 \ldots 3}, \lambda_{\text{var}}, \lambda_{\text{ani}}$ are weighting parameters. Given that the architecture is fully convolutional, this model is applicable to images with adjustable resolutions.

4. Dataset

Our dataset contains 1502 pairs of $\{$illustration $\boldsymbol{X}$, line drawing map $\boldsymbol{L}$, regular screentone segmentation map $\boldsymbol{S}$, and irregular texture mask $\boldsymbol{M}\}$ in 1024px resolution. We provide examples in the supplemental material. We invite 5 artists to annotate the dataset. The data preparation starts with a relatively small set of 56 illustration and line drawing pairs collected by searching the key word "illustration with line drawing" in Pixiv [35]. Artists align and retouch those 56 line drawings into usable annotations. Next, artists manually create 56 paired screentone segmentation maps using a commercial segmentation annotation tool [28] and 56 texture masks using Adobe Photoshop "quick select" tool. We train our framework on these initial data for 20 epochs and estimate coarse annotations for 1446 high-quality illustrations in Danbooru [12]. Artists refine those coarse estimations into final annotations. We view 100 refinements as one loop. When each loop is finished, we train our framework on the new data for 50 epochs and all old-and-new data for 20 epochs, and then re-estimate coarse annotations for the remaining

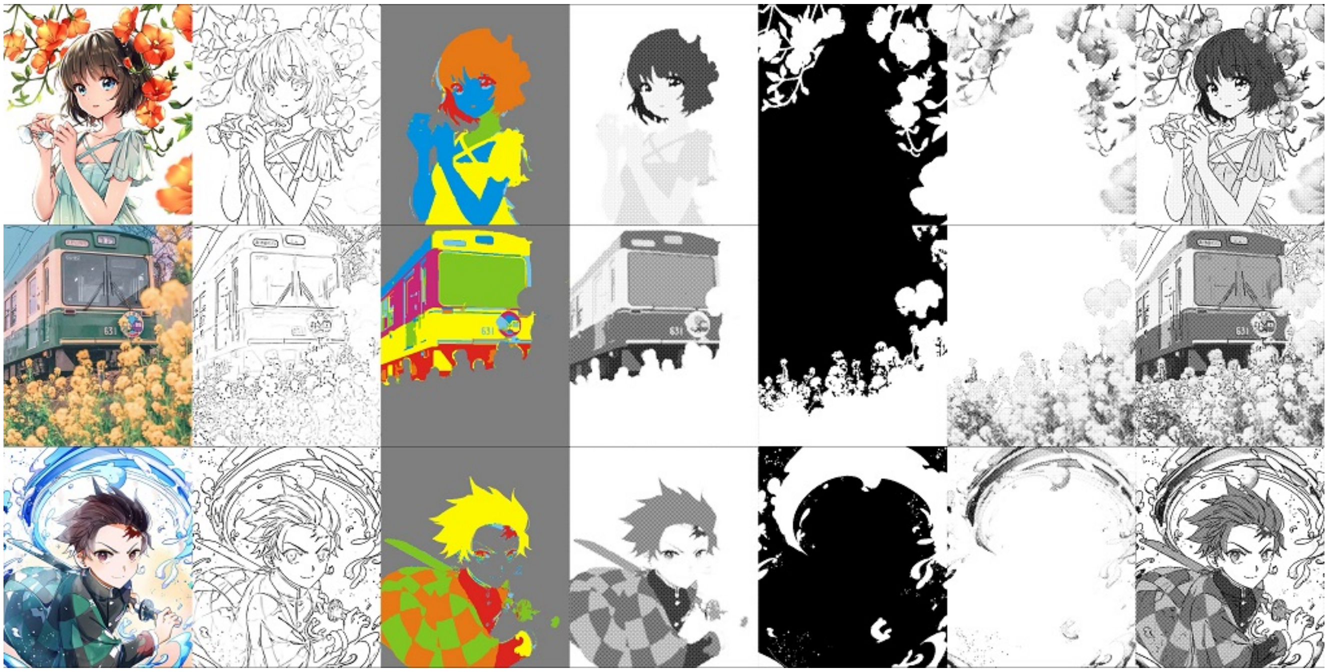


Figure 6. Qualitative results. We show the generated manga image layers. 60 additional results are available in the supplementary material.

Method	Li 2019 [26]	Qu 2008 [36]	CSP 2020 [6]	Xie 2020 [53]	Ours
AHR ↓	3.11 ± 0.53	4.66 ± 0.31	**2.66 ± 0.37**	4.11 ± 0.29	**1.33 ± 0.24**

Table 1. Average Human Ranking (AHR) of the manga images generated by different methods. The arrow (↓) indicates that lower is better. Top 1 (or 2) score is marked in blue (or red).

Method	Li 2019 [26]	Qu 2008 [36]	CSP 2020 [6]	Xie 2020 [53]	Ours
AMTFR ↑	9.52%	1.16%	**11.52%**	5.06%	**24.54%**

Table 2. Amazon Mechanical Turk Fooling Rate (AMTFR) on generated manga images. This metric reflects how indistinguishable the generated manga images are from real ones. Top 1 (or 2) score is marked in blue (or red). Higher (↑) is better.

unrefined illustrations. In parallel, all 5 artists are invited to check the annotation quality in that loop. Artists are allowed to remove any annotations when they find low-quality ones. This human-in-the-loop annotation workflow finishes when the quality of each annotation has been assured by at least three artists.

5. Evaluation

5.1. Experimental setting

Compared approaches. We test several typical manga generation methods of (1) the traditional optimization-based manga screentone method Qu *et al*. 2008 [36], (2) the commercial manga software CSP 2020 [6], (3) the learning-based stylization framework Li *et al*. 2019 [26], (4) the state-of-the-art manga effect *ScreenVAE* from Xie *et al*. 2020 [53], (5) Pix2Pix [20], and (6) our framework.

Implementation details. Our framework is trained using the Adam optimizer [23] with a learning rate of $\mathrm{lr} = 10^5$, $\beta = 0.5$, a batch size of 16, and 100 epochs. Training samples are randomly cropped to be 224×224 pixels and augmented with random left-and-right flipping. Besides, Qu [36] supports arbitrary screentone types, and we set it to use the same screentones with us. CSP [6] supports a vast majority of screentones with commercial standards, and we choose the same screentones as ours in the their interface. Li [26], Xie [53], and Pix2Pix [20] are learning-based methods with official implementations, and we train them on the *illustration-to-manga* pairs composed with our dataset.

Hyper-parameters. We use the default (and recommended) configuration of our framework: $\lambda_i = 0.5, \kappa = 0.1, \lambda_1 = 1.0, \lambda_2 = 1.0, \lambda_3 = 1.0, \lambda_{\text{var}} = 0.1$, and $\lambda_{\text{ani}} = 0.1$.

Testing samples. The tested images are Pixiv [35] and Danbooru [12] illustrations sampled randomly in different experiments. We make sure that all tested images are unseen from the training dataset.

5.2. Perceptual user study

The user study involves 15 individuals, where 10 individuals are non-artist students, and the other 5 are professional

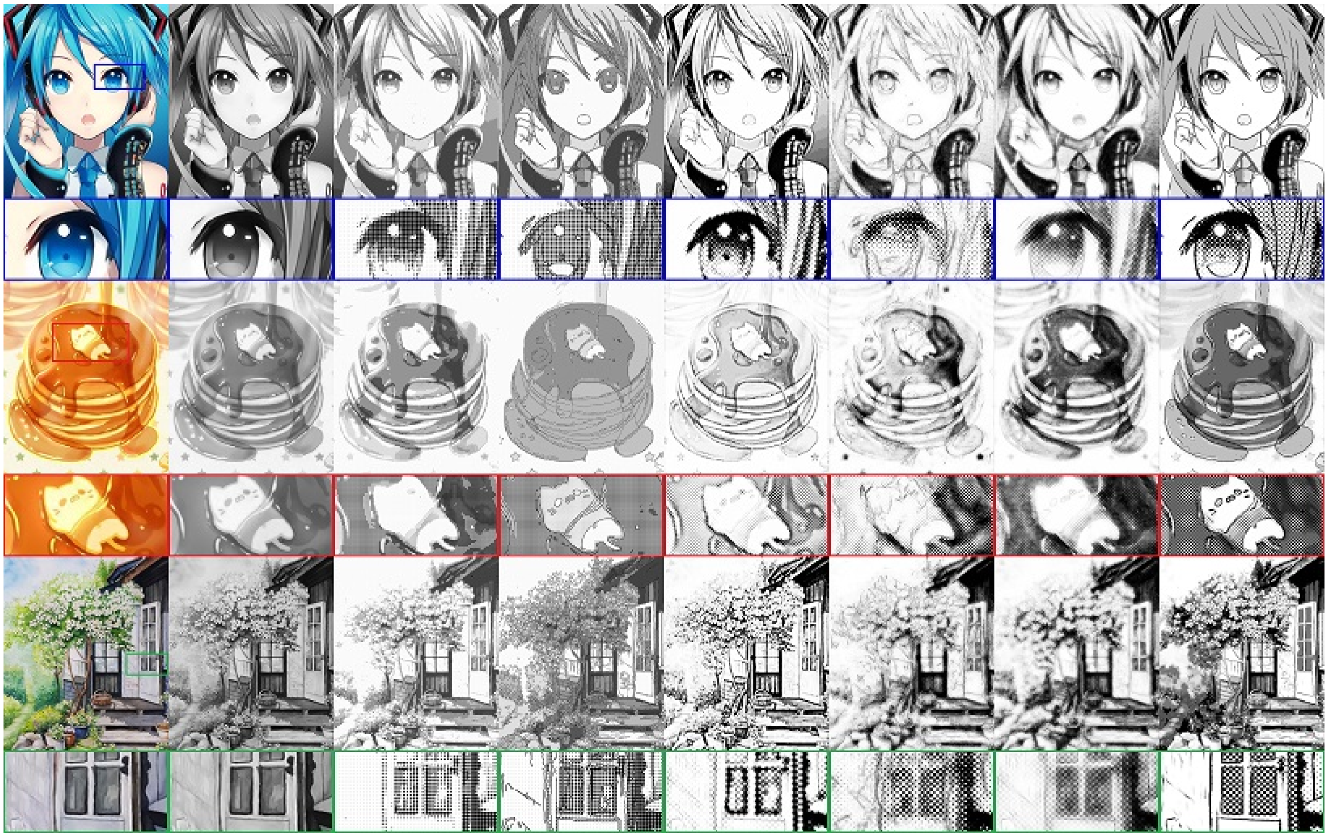


Figure 7. Comparisons to possible alternative methods. 15 additional full-resolution comparisons are available in the supplementary material.

artists. Each artist has at least three months of manga creation experience. We randomly sample 150 unseen illustrations in Danbooru [12], and then use the involved 5 methods ([53, 36, 26, 6] and ours) to generate 150 result groups, with each group containing 5 results from 5 methods. The participants are invited to rank the results in each group. When ranking the results in each group, we ask users the question – "Which of the following results do you prefer most to use in commercial manga? Please rank the following images according to your preference". We use the Average Human Ranking (AHR) as the testing metric. For each group in the 150 groups, one random user ranks the 5 results in the current group from 1 to 5 (lower is better). Afterwards, we calculate the average ranking obtained by each method.

The results of this user study is reported in Table 1. We have several interesting discoveries: (1) Our framework outperforms the secondly best method by a large margin of $1.33/5$. (2) The commercial software CSP 2020 [6]reports the secondly best score. (3) The two learning-based methods [26, 53] reports similar perceptual quality, with [53] slightly better than [26]. (4) Qu 2008 [36] reports a relatively low user preference, possibly due to its hard threshold when segmenting screentone regions.

We also conduct a user study with the Amazon Mechanical Turk Fooling Rate (AMTFR). The turk workers first watch a few real manga on the VIZ-manga [46] website, and then play with all the tested tools for 15 minutes to get familiar with the appearance of the "real/fake" manga. We randomly shuffle the 150 fake manga images (generated by the 5 tested methods as described in § 5.2) and another 150 images cropped from the real VIZ manga. We afterwards ask the turk workers to tell whether each image is real manga. The workers' mistake rate (fooling rate) reflects how the generated manga is indistinguishable from the real manga products. The result is presented in Table 2. We can see that our framework reports the highest fooling rate, more than twice that of the secondly best method. This is because our framework mimics the real manga creation workflow to simulate the manga created by artists manually.

5.3. Qualitative result

We present decomposed qualitative results in Fig. 6 and 60 additional results in the supplementary material. We can see that our framework not only generates satisfactory manga results, but also produces independent image layers of line drawing, screentone, and screen texture. These layers are

(a) Illustration (b) W/o inking (c) W/o segment (d) W/o masking (e) W/o $\phi(\cdot)$ (f) W/o $\mathcal{L}_{\text{var}}$ (g) W/o $\mathcal{L}_{\text{ani}}$ (h) Ours

Figure 8. Ablative study. We study the impact of each individual component within our framework by removing components one-by-one.

practically usable in the daily works of manga artists.

5.4. Visual comparison

We present comparisons with previous methods [26, 36, 6, 53, 20] in Fig. 7 and 15 additional comparisons in the supplementary material. We also provide greyscale images for reference. We can see that CSP 2020 [6] and Qu *et al*. 2008 [36] can only map region-wise or pixel-wise intensity to screentone dots. Li *et al*. 2019 [26] causes boundary/detail distortion, *e.g*., the girl eyes. Xie *et al*. 2020 [53] and Pix2Pix [20] suffer from severe blurring/halo artifacts (when zooming in), *e.g*., the cake decoration. Our framework mimics the artist workflow and yields sharp and clean manga products.

5.5. Ablative study

As shown in Fig. 8, the ablative study includes the following experiments. (1) We remove the inking decoder and train our framework without line drawing maps. We can see that the removal of line drawing fails our framework in outlining the object structure (Fig. 8-(b)). (2) If trained without the screentone segmentations (with the segment decoder removed), the framework cannot mimics the artist behaviour of region-by-region screentone pasting, yielding screentone distortions (Fig. 8-(c)). (3) If trained without the screen texture masks (with the mask decoder removed), the framework fails to capture appropriate textures, resulting in dull and defective tone transitions (Fig. 8-(d)). (4) If trained without the line pattern penalty $\phi(\cdot)$, the lines become blurred (Fig. 8-(e)). (5) If trained without the region-wise variation penalty $\mathcal{L}_{\text{var}}$, the framework suffer from screentone type inconsistency (Fig. 8-(f)). (6) If trained without the anisotropic penalty $\mathcal{L}_{\text{ani}}$, the textured areas become uncontrolled and noisy (Fig. 8-(g)). (7) The full framework suppresses these types of artifacts and achieves a satisfactory balance over the line drawing, screentone, and screen texture (Fig. 8-(h)).

5.6. Influence of hyper-parameters

A weak region-wise variation penalty ($\mathcal{L}_{\text{var}} = 0.01$) causes inconsistency tone over adjacent regions, whereas a too strong penalty ($\mathcal{L}_{\text{var}} = 1.0$) causes texture vanishing. Besides, a weak anisotropic penalty ($\mathcal{L}_{\text{ani}} = 0.01$) causes textural distortion, whereas a too strong penalty ($\mathcal{L}_{\text{ani}} = 1.0$) causes low contrast in detailed constituents. See also the supplement for examples.

6. Conclusion

We present a framework to generate manga from illustrations by mimicking the manga creation workflow, including the steps of line drawing inking, regular screentone pasting, and irregular screen texture pasting. We invite artists to annotate a large-scale dataset to train the neural networks, and the dataset will be publicly available. Both quantitative and qualitative experiments elaborate that the users prefer our layered manga products compared to possible alternatives.

7. 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). We thank Qianwen Lu for the helpful discussions for the technical merits and limitations.

References

[1] Adobe. Adobe after effects. *Adobe*, 2020.

[2] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CVPR*, 2017.

[3] J. W. BUCHANAN, , VEREVKA, and O. Edge preservation with space-filling curve half-toning. In *Proceedings of Graphics Interface 95, 7582*. ACM, 1995.

[4] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017.

[5] 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.

[6] clip studio. Clip studio paints. *clip studio*, 2020.

[7] clip studio. The complete clip studio paint screentone tutorial. *https://doncorgi.com/blog/clip-studio-paint-screentone-tutorial/*, 2020.

[8] clip studio. Csp: Turning your art into screentone. *https://digiartstuff.wordpress.com/2019/04/18/csp-turning-your-art-into-screentone/*, 2020.

[9] clip studio. Making comic backgrounds using layer properties. *https://tips.clip-studio.com/en-us/articles/1164*, 2020.

[10] clip studio. Turning color illustrations into black and white manga. *https://tips.clip-studio.com/en-us/articles/1339*, 2020.

[11] clip studio manga materials. Convert image to comic style background bacics of photo lt conversion. *https://manga-materials.net/en/2dlt_b/*, 2020.

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

[13] DELETER. Deleter official shopping site. *http://deleter-mangashop.com/*, 2020.

[14] Jeff Donahue, Philipp Krhenbhl, and Trevor Darrell. Adversarial feature learning. *ICLR*, 2017.

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

[16] R. W. FLOYD, , STEINBERG, and L. An adaptive algorithm for spatial grey scale. In *SID International Symposium Digest of Technical Papers, Society for Information Display, 3637*. ACM, 1974.

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

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.

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

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[21] J. F. JARVIS, C. N. JUDICE, , NINKE, and W. H. A survey of techniques for the display of continuous tone pictures on bilevel displays. *Comput Graphics Image Process 5*, 1, 1976.

[22] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *ICML*, 2017.

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

[24] D. E. KNUTH. Digital halftones by dot-diffusion. *ACM Transactions on Graphics 6*, 4, 1987.

[25] Chengze Li, Xueting Liu, and Tien-Tsin Wong. Deep extraction of manga structural lines. *ACM Transactions on Graphics*, 36(4), 2017.

[26] 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.

[27] 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.

[28] lionbridge. Lionbridge data annotation platform. *https://lionbridge.ai/data-annotation-platform*, 2020.

[29] Chenxi Liu, Enrique Rosales, and Alla Sheffer. Strokeaggregator: Consolidating raw sketches into artist-intended curve drawings. *ACM Transactions on Graphics*, 2018.

[30] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *NIPS*, 2017.

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

[32] A. NAIMAN, , LAM, and D. Error diffusion: wavefront traversal and contrast considerations. In *Proceedings of Graphics Interface 96, 7886*. ACM, 1996.

[33] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. *ICML*, pages 807–814, 2010.

[34] Wai-Man Pang, Yingge Qu, Tien-Tsin Wong, Daniel Cohen-Or, and Pheng-Ann Heng. Structure-aware halftoning. *ACM Transactions on Graphics (SIGGRAPH 2008 issue)*, 27(3):89:1–89:8, 2008.

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

[36] Yingge Qu, Wai-Man Pang, Tien-Tsin Wong, and Pheng-Ann Heng. Richness-preserving manga screening. *ACM Transactions on Graphics (SIGGRAPH Asia 2008 issue)*, 27(5):155:1–155:8, December 2008.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015.

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

[39] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Real-time data-driven interactive rough sketch inking. *ACM Transactions on Graphics*, 2018.

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

[41] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Sketching cartoons by example. In *Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 27–34, 2005.

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

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

[44] R. A. ULICHNEY. Digital halftoning. *MIT Press*, 25(4), 1991.

[45] L. VELHO, , GOMES, and J. M. *Stochastic screening dithering with adaptive clustering*. Computer Graphics (Proceedings of SIGGRAPH95), 273276, 1994.

[46] viz. Viz manga. *https://www.viz.com/read*, 2020.

[47] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *CVPR*, 2018.

[48] 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.

[49] wikipedia. Softmax function. *wikipedia*, 2020.

[50] G. WINKENBACH, , SALESIN, and D. H. *Computergenerated pen-and-ink illustration*. Proceedings of SIGGRAPH 1994, 91100, 1994.

[51] Tien-Tsin Wong. The modelling of natural imperfections and an improved space filling curve halftoning technique. *Master thesis The Chinese University of Hong Kong*, 1994.

[52] Tien-Tsin Wong and Siu chi Hsu. Halftoning with selective preciptation and adaptive clustering. *Graphics Gems V*, 1995.

[53] Minshan Xie, Chengze Li, Xueting Liu, and Tien-Tsin Wong. Manga filling style conversion with screentone variational autoencoder. *ACM Transactions on Graphics (SIGGRAPH Asia 2020 issue)*, 39(6):226:1–226:15, December 2020.

[54] 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.

[55] 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.

[56] 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.

[57] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. *ICCV*, 2017.

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

[59] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *ICCV*, 2017.