Deep Extraction of Manga Structural Lines

CHENGZE LI, XUETING LIU, TIEN-TSIN WONG, Department of Computer Science and Engineering,The Chinese University of Hong Kong and Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

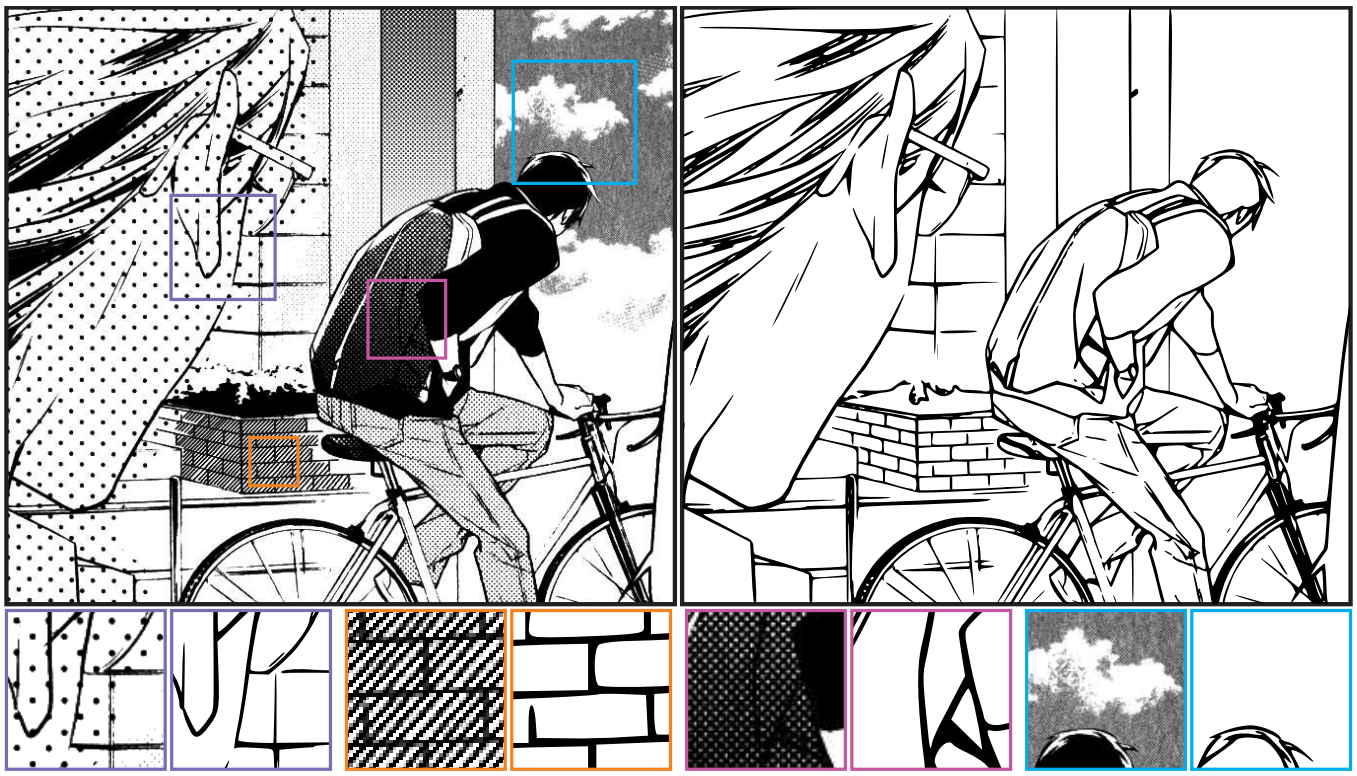


Fig. 1. Our method automatically extracts structural lines and removes textures from manga images. ©Takarai Rihito

Extraction of structural lines from pattern-rich manga is a crucial step for migrating legacy manga to digital domain. Unfortunately, it is very challenging to distinguish structural lines from arbitrary, highly-structured, and black-and-white screen patterns. In this paper, we present a novel data-driven approach to identify structural lines out of pattern-rich manga, with no assumption on the patterns. The method is based on convolutional neural networks. To suit our purpose, we propose a deep network model to handle the large variety of screen patterns and raise output accuracy. We also develop an efficient and effective way to generate a rich set of training data pairs. Our method suppresses arbitrary screen patterns no matter whether these patterns are regular, irregular, tone-varying, or even pictorial, and regardless of their scales. It outputs clear and smooth structural lines even if these lines are contaminated by and immersed in complex patterns. We have evaluated our method on a large number of mangas of various drawing styles. Our method substantially outperforms state-of-the-art methods in terms of visual quality. We also demonstrate its potential in various manga applications, including manga colorization, manga retargeting, and 2.5D manga generation.

CCS Concepts: • **Applied computing → Arts and humanities**; *Fine arts*;

Additional Key Words and Phrases: Computational manga, structural line extraction

ACM Reference format:
Chengze Li, Xueting Liu, Tien-Tsin Wong. 2017. Deep Extraction of Manga Structural Lines. *ACM Trans. Graph.* 36, 4, Article 117 (July 2017), 12 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073675

This work is supported by Shenzhen Science and Technology Program (No.JCYJ20160429190300857), and Research Grants Council of the Hong Kong Special Administrative Region, under RGC General Research Fund (Project No. 14200915).

DOI: http://dx.doi.org/10.1145/3072959.3073675

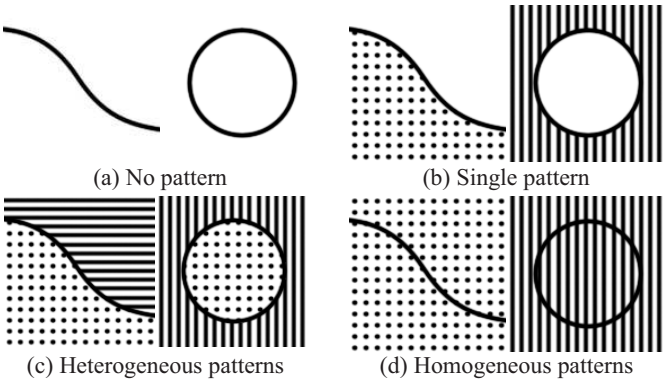


Fig. 2. Possible combinations of screening on the two sides of the structural line.

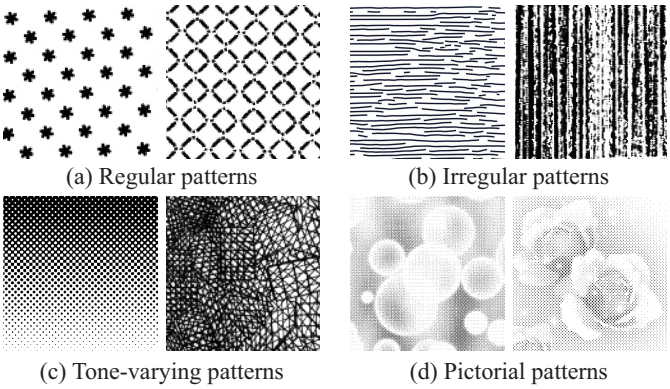


Fig. 3. Various types of patterns.

1 INTRODUCTION

With the wide popularity of portable devices and the low-cost distribution over the internet, there has been an increasing trend to convert legacy manga to digital form. Comparing to traditional paper-based manga (typically printed in black-and-white for cost saving), electronic manga or e-manga is more visually appealing as more visual elements, such as color presentation and powerpoint-like animation, can be easily introduced. With vectorization, e-manga may even be retargeted to the desired resolution on the portable device for optimal display, and free of over-blurring or aliasing.

Manga digitization usually starts with scanning and continues with various semantic processing procedures, such as segmentation, object layer extraction, or structural-line and screen-pattern separation. While scanning mangas to raster images is trivial, the subsequent semantic processing procedures are much more challenging. Most of these semantic processing procedures require the identification/extraction of structural lines. Here, the structural lines may be exterior boundaries of white or screen pattern regions (e.g. blue box in Fig. 1 and Fig. 2(a)-(c)) or interior lines immersed in screen patterns for decoration/structure depiction (e.g. orange box in Fig. 1, and Fig. 2(d)). Furthermore, the screen patterns are black-and-white, and can be of *arbitrary* types and styles. They can be regular patterns (Fig. 3(a)), irregular patterns (Fig. 3(b)), tone-varying patterns (Fig. 3(c)), or even pictorial patterns (Fig. 3(d)). They can be composed of lines, just like the structural lines we aim for (orange box in Fig. 1 & 4(a)). This is also why high-quality tracing of structural lines is still done by hand in the current manga industry, which is labor-intensive and time-consuming.

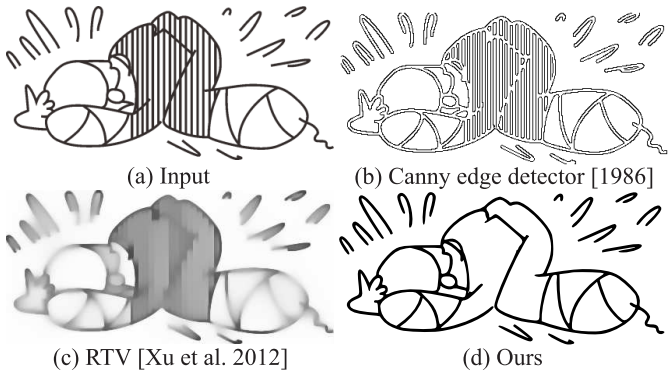


Fig. 4. Comparison of existing line extraction methods.

To automate the extraction of structural lines, one may apply existing edge detection techniques [Canny 1986; Ito et al. 2015; Kang et al. 2007]. However, these methods easily get confused by the textural lines of screen patterns commonly used in manga (e.g. dress of the person in Fig. 4(a)) and may also result in double-edge artifacts (Fig. 4(b)). Texture removal and texture smoothing techniques can also be applied [Tomasi and Manduchi 1998; Vese and Osher 2003; Xu et al. 2012]. But most of them rely on local oscillations, so they are only good for suppressing small-scale textures and fail to suppress large-scale textures that exhibit similar features as the targeting structural lines (Fig. 4(c)). While several manga-tailored screen pattern extraction methods have also been proposed, they either have strong assumptions on the types of the screen patterns [Kopf and Lischinski 2012; Yao et al. 2017] or rely on user guidance [Qu et al. 2006].

Our goal is to extract structural lines from manga laid with arbitrary screen patterns. Due to the wide variety of screen patterns, extracting structural lines from arbitrary screen patterns is a hard recognition problem. Learning-based method has recently been demonstrated to be effective in solving several other image recognition problems. In this paper, we propose an automatic data-driven method based on convolutional neural networks (CNN) to tackle this challenging structural line extraction problem. We demonstrate that, by properly designing the network structure of the CNN and providing sufficient training data for training the CNN, we can effectively and efficiently extract the structural lines immersed in arbitrary screen patterns (Fig. 1).

To achieve that, we develop a rich dataset that contains a large number of training pairs, containing rich styles of screen patterns as well as different styles of structural lines. As manual labeling is very tedious and labor-intensive, we also propose an automatic method to generate the training data pairs. To apply CNN to our application, we propose a relatively deeper network structure to handle the large variety of screen patterns. We further improve

our network by borrowing the ideas from the recent advances such as residual network [He et al. 2016a] and symmetric skipping network [Ronneberger et al. 2015] to increase output accuracy and robustness to high-frequency components. Through feeding with the training data, the system learns a sequence of convolutional operators in multiple scales for structural line extraction purpose. After the offline training, the online structural line extraction is simply a set of convolutional operations with trained filter weights. It takes the manga image as input, and generates an output image containing only the structural lines, with all screen patterns being filtered away.

To validate the effectiveness of our method, we test our method with a rich variety of challenging cases. Convincing and clear structural lines are obtained. Fig. 1 shows our result of a challenging real-world example that fails most existing methods. We also compare our results to those of existing methods to demonstrate our robustness in handling arbitrary screen patterns. In addition, we further demonstrate how to make use of our clear structural lines for multiple interesting applications, including manga colorization, manga retargeting, and 2.5D manga. Our contributions can be summarized as followed.

- We propose a novel convolutional neural network framework tailored for extracting structural lines from manga with arbitrary screen patterns.
- We propose an effective way to automatically generate a rich set of training data.
- We demonstrate multiple potential applications to utilize the clear structural lines resulted from our extraction engine.

2 RELATED WORKS

Before introducing our method, we first review existing works on line extraction. They can be roughly classified into three categories, namely edge detection methods, texture removal methods, and CNN-based techniques.

Edge Detection Edge detection is a classical research problem. Many edge detectors have been proposed, such as Canny edge detector [Canny 1986], Laplacian of Gaussians (LoG), and flow-based difference of Gaussians [Kang et al. 2007]. However, they all heavily rely on the gradient, and are easily confused by the black-and-white high-contrast screen patterns. Attempts have also been made in boundary detection for natural images [Arbelaez et al. 2011; Isola et al. 2014; Kokkinos 2015]. While these methods are tailored for detecting overall structure of objects, they generally perform poorly in extracting fine structural lines in manga images, especially when the lines are immersed in screen pattern. In particular, Isola et al. [2014] proposed to extract crisp-boundary by learning the pairwise pixel dependency of image features and grouping the pixel affinity function. Their method achieves reasonable results for natural images, but fails to identify the structural lines immersed in screen patterns due to the high similarity of local statistics.

Several cartoon-tailored line extraction methods have been proposed based on edge detection. Sýkora et al. [2004] proposed to extract potential edge pixels by applying the LoG filter followed by a negative mask. Zhang et al. [2009] proposed to combine Canny edge detector and Steger's linking algorithm to detect decorative lines in cartoon animations. Liu et al. [2013] proposed to extract the edges by applying adaptive histogram equalization followed by a median filtering. Mao et al. [2015] further proposed a region-based approach to classify edge regions and non-edge regions. Note that all above methods are mainly designed for cartoons that typically contain significantly fewer screen patterns. In sharp contrast, bitonal manga images are usually dressed up with screen patterns to compensate its lack of colors. These high-contrast patterns usually fail those cartoon-tailored methods. Ito et al. [2015] combined existing edge detectors and suppressed high-frequency noise to separate lines from screen patterns. However, it still relies on gradients and may easily fail when the structural line is immersed in screen patterns. In contrast, our learning-based method does not solely rely on gradient information.

Texture Removal Attempts have also been made in identifying structural lines by removing textures. For regular or near regular patterns in natural images, several approaches have been proposed [Hays et al. 2006; Liu et al. 2008, 2004a,b], but they may fail on arbitrary screen patterns. To handle arbitrary textures, texture smoothing techniques have been proposed based on local oscillations, such as bilateral filtering [Tomasi and Manduchi 1998], total variation regularization [Vese and Osher 2003], L_0 gradient optimization [Xu et al. 2011] or relative total variations [Xu et al. 2012]. These methods can also be used to differentiate structures from screen patterns, but are usually quite limited to small-scale patterns due to their local nature. Unfortunately, manga images usually contain large-scale screen patterns such as sparse grids or stripes that can easily fail these methods, as demonstrated in Fig. 4(c).

To identify the screen patterns, multiple manga-tailored methods have been proposed. Qu et al. [2006] proposed to segment screened regions via level-set by analyzing its Gabor wavelet features. While texture-analysis based methods above can recognize arbitrary screen patterns, they fail to precisely identify the boundary as texture features change abruptly near the boundary. The resultant boundaries are usually very noisy or rough. Interior structural lines are usually missed if the texture features of these lines are very similar to that of the surrounding screen patterns. In contrast, our learning-based method can accurately identify both exterior boundaries or interior structural lines.

Kopf et al. [2012] proposed to reconstruct and vectorize halftoned comics by modeling the dotted patterns. Yao et al. [2017] proposed to analyze and recognize three types of screen patterns: dots, stripes, and grids. If the target pattern falls into their modeled ones, their results should be very nice. However, real-world screen patterns usually do not fit well into their limited number of screen pattern models, and hence fail their methods. In contrast, our learning-based approach makes no assumption of the types of screen patterns.

Convolutional Neural Networks Based on deep neural networks, several edge extraction methods have been proposed recently, such as HED [Xie and Tu 2015] and DeepContour [Shen et al. 2015]. HED employs a multi-scale feature extraction network and generates holistic edge map from the fusion of network outputs with different reception fields. DeepContour subdivides edges into different classes and processes them with different network parameters.

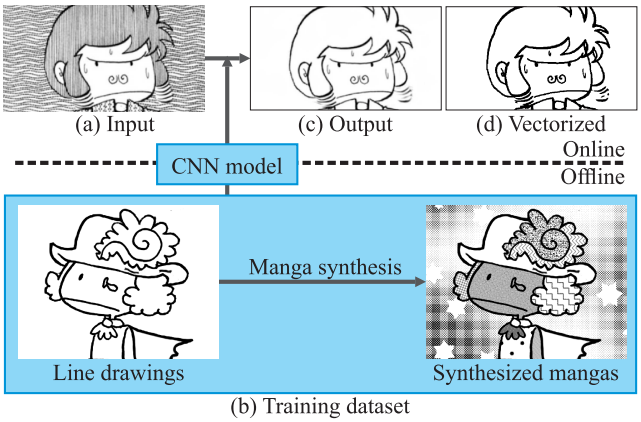


Fig. 5. System overview. ©Minamoto Tarou

While both methods generate reasonable results when applied on natural images, they may incorrectly identify textural lines of the screen pattern (such as the dress in Fig. 4(a)) as structural lines when applied on manga. This is because that screen patterns in manga may exhibit similar line features as structural lines we aim for. The CNN structures and training data of the existing works are not designed to differentiate the small semantic difference between large-scale screen patterns and structural lines. In this paper, we design a novel CNN structure and prepare the training data specifically for our manga structural line extraction purpose. In particular, our design pays extra attention on the quality of the extracted lines, to counteract the blurry or noisy edges.

Simo-Serra et al. [2016] proposed a CNN-based sketch simplification method. They optimized the network structure and trained it with a dataset tailored for their sketch simplification application. Since the CNN structure is tailored for line drawings, their method cannot be directly applied to our application as manga images with arbitrary screen patterns are usually more complex than line drawings. In our application, a deeper network structure is needed to overcome the overcrowded details in manga, while remains able to recognize the pattern-immersed structural lines.

3 OVERVIEW

Fig. 5 illustrates the major processes in our system. It involves an offline training phase and an online phase of structural line extraction. During the online phase, our system takes an arbitrary input manga (Fig. 5(a)), and generates an image containing mainly the structural lines with screen patterns removed (Fig. 5(c)). Both input and output images have the same resolution and are in grayscale. The reason we allow input to be grayscale is to take care those manga images that are scanned in lower resolution (bitonal content scanned as blurry gray values). The computation is mainly a series of convolutional operations that can be rapidly performed on modern GPU.

The weights used in convolutional operations are the training results obtained from CNN model training. This training is done only once and offline (Fig. 5(b)). During the training phase, a large-scale training dataset is needed to train the network for better performance. However, preparing thousands of training pairs of manga images and clear structural lines images is not trivial, as manually tracing structural lines from manga is extremely tedious and time-consuming. Instead of tracing structural lines from real-world manga, we can generate screened manga images from clear line drawings (without any screen pattern) by laying a rich library of screen patterns, as illustrated in the Fig. 5(b). The details of preparing the training dataset and the design of CNN are described in Section 4.

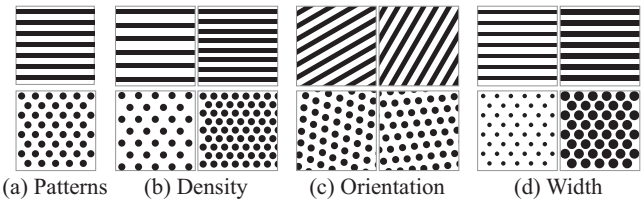


Fig. 6. Variants of screen patterns.

Once the structural line image is generated, it can be used for the subsequent applications. Depending on the application, the structural line image can be used as guidance map to guide the processing on the input manga, or directly used for image creation. In both cases, the structural line image is first binarized with a fixed threshold $T_b = 225$ in which the pixel value range is [0,255]. Here, we use a relatively high threshold for binarization and vectorization to preserve low-gradient structural lines. Vectorization of this structural line image is optional. It is especially needed if the application changes the resolution. Detail description on each application can be found in Section 6.

4 LEARNING-BASED STRUCTURAL LINE EXTRACTION

4.1 Training Data Preparation

A key to the success of CNN-based methods is the variety/richness of the training data. This also means that a large number of training data is needed for good performance. Each training data pair should contain a screen-rich manga and its corresponding screen-free counterpart. Asking experienced artists to trace structural lines from manga images is extremely labor-intensive, time-consuming, and costly. Besides, manual tracing is basically a re-drawing process of the structural lines. Both the location (e.g. misalignment) and the style (e.g. different line width) may not be exactly the same as that in the original manga. This is not desirable in training. In this paper, we propose to automatically generate a large-scale training dataset. Instead of tediously tracing structural lines from existing manga, we adopt a reverse process to synthesize the screen-rich manga from the screen-free line drawings, by laying a rich library of screen patterns. With this approach, we avoid the tedious tracing, misalignment, or line style inconsistency issues mentioned above.

To generate the training dataset, we start with 117 bitonal line drawings of various styles drawn by different artists, and 110 commonly used screen patterns from existing mangas in the Manga109 database [Matsui et al. 2015]. These screen patterns include 36 regular patterns, 24 irregular patterns, 27 tone-varying textures, and 23 pictorial patterns (see Fig. 3 for some examples). To enrich the screen pattern library, for each type of screen pattern, we further collect or create multiple variants by varying its density (Fig. 6(b)), orientation (Fig. 6(c)), and line width (Fig. 6(d)). This results in 56,700 various

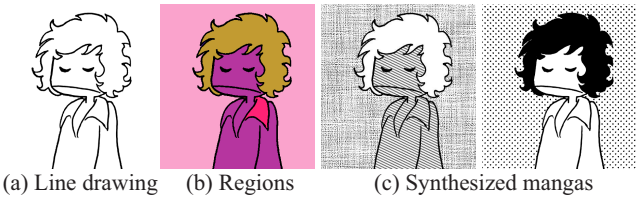


Fig. 7. Generating screen-rich mangas from line drawings. ©Minamoto Tarou

screen patterns in total. Readers are referred to the supplementary materials for a more detailed listing of sample screen patterns and line drawings.

Next, we generate the screen-rich mangas by laying various combination of screen patterns over the line drawings. Before that, we need to segment the regions in the line drawing, as manga artists typically fill up the regions with screen patterns. Instead of sophisticated segmentation technique, a simple flood-filling is sufficient to serve our purpose (see Fig. 7(b) for these regions). At first glance, it seems that we can generate screen-rich manga by filling a randomly selected region with a screen pattern randomly selected from the library. If we do so, the trained CNN model will be contaminated by unrealistic manga images that may not exist in the real world. Imagine that random selection can fill a tiny region with large-scale screen patterns or pictorial patterns. Readers (as well as our CNN model) can no longer differentiate whether the filled pattern is a screen pattern to ignore or fine details to preserve (see Fig. 8(b) for example). Moreover, we also found that CNN model trained with random screening is very poor and extracts noisy and broken structural lines. We believe manga artists understand that some rules are required to avoid the resultant manga confusing the readers, and hence they intentionally avoid such undesirable screening. So we mimic manga artists in screening a region, by following the rules below:

- *Rule 1:* Larger regions tend to be filled with larger-scale screen patterns, and vise versa (Fig. 8(a)-(c)).
- *Rule 2:* Larger regions are more likely to be filled with tone-varying/pictorial patterns (Fig. 8(d)-(f)).
- *Rule 3:* Solid black is also regarded as a type of screen pattern. It will not be selected for region containing interior structural lines. Otherwise, interior structural lines will become no longer observable (Fig. 8(g)-(i)). Besides, two neighboring regions cannot be filled with solid black simultaneously (Fig. 8(j)-(l)).
- *Rule 4:* A region may be filled with screen patterns or left solid white. We leave 20% of the regions to be solid white.

Fig. 7(c) shows some generated screen-rich mangas. They may not be the same as the real-world manga as our generator lacks of semantic understanding of the drawing, but these mangas are good enough for training purpose.

To train the CNN model to tolerate different scanning resolutions of the input manga, we resample each training pair to multiple resolutions by downscaling the images into 7/8, 3/4, 5/8 and 1/2 of the original dimensions. During the training, we extract 512×512 patches cropped from the screen-rich manga as training inputs, and the corresponding 512×512 patches cropped from the screen-free

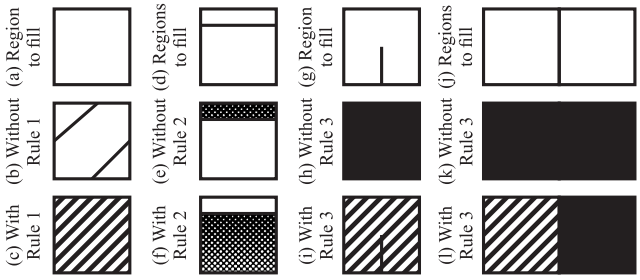


Fig. 8. (a) A tiny region. (b) Without Rule 1, we may fill region (a) with large-scale patterns and lead to ambiguity. (c) With Rule 1, small regions is only filled with small-scale patterns. (d) Two regions of different areas. (e) Without Rule 3, we may fill the small region with unrecognizable tone-varying patterns. (f) Tone-varying patterns can only be used to fill large regions. (g) A region with an interior structural line. (h) Without Rule 3, we may fill (g) with solid black and completely hide the interior line. (i) With Rule 3, we avoid such hiding. (j) Two neighboring regions. (k) Without Rule 3, both regions may be filled with solid black and hide the in-between structural line. (l) With Rule 3, such hiding is also avoided.

line drawing as training outputs. We also randomly flip the patches horizontally to reduce overfitting.

4.2 CNN Model

In designing an appropriate CNN model, we have two requirements. Firstly, the extracted structural lines need to be aligned to those of the input in a pixel level. Misalignment or line thickening/thinning should be minimized. Secondly, as the input is in gray level (depending on the scanning resolution), it is better to preserve the pixel intensity of the structural lines in the output, without introducing extra information or removing the original information of the structural lines. To achieve these, a pixel-wise CNN model [Noh et al. 2015; Simo-Serra et al. 2016] is more desirable.

A typical pixel-wise CNN model is composed of two parts: an encoding network and a decoding network. The encoding network is inspired by traditional classification networks [Krizhevsky et al. 2012; Simonyan and Zisserman 2014], and compresses the input into feature vectors. This feature extraction process is usually modeled by a sequence of blocks consisting of three types of layers including convolutional layers, activation layers, and pooling layers. The pooling layers with strides are regularly used to reduce the dimension of the input. The decoding network is built after the encoding network. It is used to reconstruct the desired output from the encoded feature vectors. Similar to the encoding network, the decoding network is also modeled by a sequence of blocks containing three types of layers including convolutional layers, activation layers, and unpooling layers. Here, unpooling layers are used to resample the feature vectors into the original image resolution. Fig. 9 illustrates our proposed network structure. It has a downscaling-upscaling structure. The left half is the downscaling network while the right half is the upscaling network.

Downscaling and Upscaling Networks The downscaling network extracts image features via a sequence of downscaling blocks and regular blocks in multiple levels (left part of Fig. 9). Blocks with the same size are on the same level. In particular, each level consists of one downscaling block followed by a sequence of regular

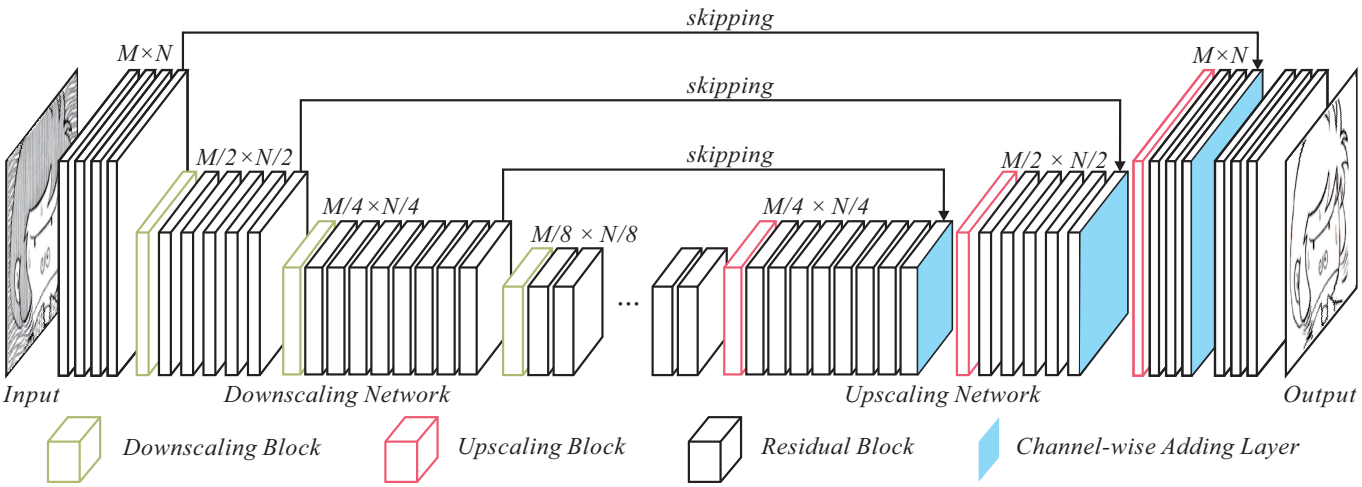


Fig. 9. Overall structure of the proposed network. ©Minamoto Tarou

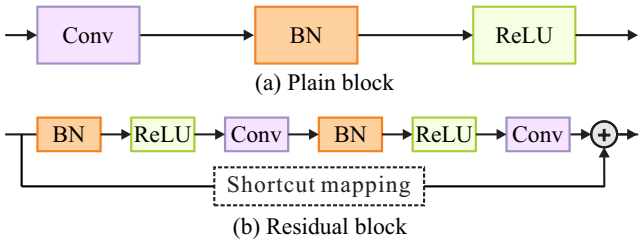


Fig. 10. Plain block structure vs. residual block structure.

blocks to increase the network depth. Instead of downscaling to 1×1 feature vectors, we propose to have 3 levels of downscaling to avoid information loss. We found that 3 levels of downscaling is already sufficient to filter away most of the textural components of various scales. Further increasing the downscaling levels does not improve the result, but unnecessarily filters away important structural components and leads to blurry structural lines in the output. On the other hand, reducing the number of downscaling levels to 1 or 2 cannot filter away the textural pattern due to the limited receptive field. So a downscaling network of 3 levels is optimal for our application (left part of Fig. 9).

While the downscaling network produces rough approximations of the structural lines in coarser scales, the upscaling network deconvolves the rough approximation and reconstructs the output from coarse to fine. In our model, the upscaling network also contains 3 upscaling levels (right part of Fig. 9) corresponding to the downscaling network. Each level is composed of one upscaling block followed by a sequence of regular blocks. Note that, the number of upscaling blocks and the number of downscaling blocks must be the same to ensure that the output has the same resolution as the input.

Residual Network Structure The basic block used in typical CNN model is a plain block consisting of a convolutional layer *Conv* followed by a batch normalization layer *BN* [Ioffe and Szegedy 2015] and a rectified linear unit layer *ReLU* as shown in Fig. 10(a). Here, a block can be downscaling, upscaling, or regular, and the corresponding convolutional layer *Conv* in the block may be downscaling, upscaling, or flat convolutions as defined in Simo-Serra et al. [2016]. However, this basic block *Conv-BN-ReLU* is too "plain" and may lead to degrading problem due to its streamlined nature, as gradient is hard to propagate from higher levels to lower levels. It may be unable to produce high-quality results when the input has a complex content, such as the screen patterns in our manga application, even with deeper levels of convolutions. To increase the depth of the network while avoiding this degrading problem, we adopt the residual block [He et al. 2016b] in our CNN model. This residual block performs the *BN-ReLU-Conv* structure twice as the main streamline where the first convolutional layer may be downscaling, upscaling, or flat convolutions. Its major difference is the introduction of a shortcut mapping directly from the input to the output to preserve important information of the input (Fig. 10(b)). Through the shortcut mapping, the input bypasses the streamline processing and directly adds to the output channel by channel. In particular, for regular blocks where the input and the output are in the same dimension, the shortcut mapping is an identity matrix. For upscaling and downscaling blocks where the input and the output are in different dimensions, the shortcut mapping is a dimension adaptation convolution which adapts the dimension of the input to fit the dimension of the output. By replacing the basic blocks with the residual blocks, it allows direct information propagation between the input and the output. Such deeper network structure leads to easier training and better line extraction quality.

Convolution/Deconvolution Layers with Strides Typical max-pooling used in CNN model is a parameter-fixed function and generally leads to loss of spatial relation in every 2×2 downscaling window during the downscaling. Unfortunately, manga images usually contain subtle high-frequency details, such as repetitive 2×2 dots (Fig. 11(a)). In this case, max-pooling simply takes the local maximal value in every 2×2 window (which is white) and outputs an image with crucial information loss (Fig. 11(b)). The information loss further affects the output quality. To preserve spatial relation and image details, instead of using the standard pooling/unpooling for

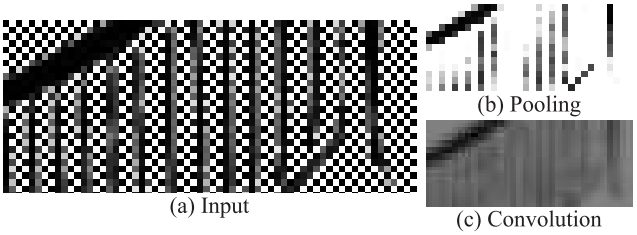


Fig. 11. Downscaling by different methods. (b) Max-pooling breaks the spatial continuity. (c) Convolution achieves better downscaling results.

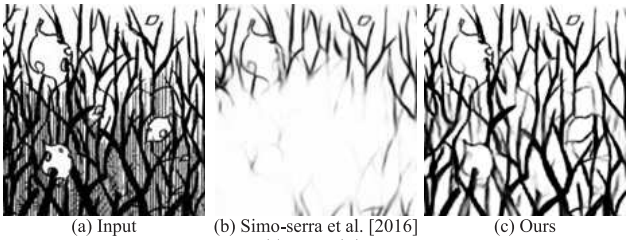


Fig. 12. Information loss in plain networks. ©Minamoto Tarou

Name	Type	Output Size	Filters × Rpt.	Parents
input	Input	$M \times N$		-
down1	Downscaling	$M \times N$	32×4	input
down2	Downscaling	$M/2 \times N/2$	64×6	down1
down3	Downscaling	$M/4 \times N/4$	128×9	down2
flat1	Flat	$M/8 \times N/8$	256×12	down3
up1	Upscaling	$M/4 \times N/4$	128×9	flat1
skip1	Merging	$M/4 \times N/4$	128	up1 down3
up2	Upscaling	$M/2 \times N/2$	64×6	skip1
skip2	Merging	$M/2 \times N/2$	64	up2 down2
up3	Upscaling	$M \times N$	32×4	skip2
skip3	Merging	$M \times N$	32	up3 down1
flat2	Flat	$M \times N$	16×3	skip3
output	Flat	$M \times N$	1×1	flat2

Table 1. The network structure. All downscaling, upscaling, and flat layers are residual-based.

downscaling/upscaling, we adopt the convolutions/deconvolutions with strides [Springenberg et al. 2014] in our downscaling/ upscaling residual blocks (Fig. 11(c)). With this convolutional downscaling/upscaling layers, the parameterized kernels are learnable during the training process. Therefore, the details of the input are more likely to be preserved by the downscaling network, and this significantly improves the quality of the extracted structural lines.

Skipping Structures Inspired by U-Net [Ronneberger et al. 2015], we create a symmetric skipping between layers of the same scale to further reduce the information loss caused by the dimension reduction. With skipping layers, image information (e.g. gradient) at certain level in the downscaling network can be directly passed to the layer of the same level in the upscaling network without compression. This means that details of the input can be better preserved, and hence leads to better output accuracy. To do so, for each level in the upscaling network, we add a merging layer as the final layer of the level (blue layer in Fig. 9), which sums the skipped image and the sequentially processed image in a channel-wise manner. The merged results are then used for next upper level of upscaling.

Table 1 lists our network structure in detail. For the ease of implementation, we pad the width M and height N of the input image to multiple of 16 to make sure the input of the merge layers has identical dimensions. Fig. 12 presents a visual comparison between the typical "plain" network model and ours.

4.3 Learning and Optimization

During training, a variety of similarity measurements can be used as the objective function to measure the objective loss of training examples. Among the similarity metrics, structural similarity index (SSIM) along with mean absolute error (MAE) is the most appropriate for our case [Zhao et al. 2016]. However, SSIM is computationally demanding for our training usage. MAE tends to search for the median and thus generates "binarized" results, so the tones of the structural lines are generally lost with the MAE metric. Instead, we adopt the mean square error (MSE) as the measurement metric since smoother structural lines can be obtained with nearly accurate intensity and pressure.

We train the model with the ADAM solver [Kingma and Ba 2014] (learning rate = 0.001, $\beta = 0.9$) against the objective function from scratch. The solver adaptively adjusts the learning rate for faster convergence. The training process is terminated when the objective loss stops decreasing.

5 RESULTS AND DISCUSSIONS

To evaluate our method, we test it on several manga images of different styles, including western (Fig. 18(g)) and Japanese (Fig. 1, 13(a)-(d), and 18(a)-(f)) styles. More results can be found in the supplementary materials. Fig. 18 compares our method to multiple state-of-the-art methods in terms of the ability to extract structural lines from regular patterns (dots in Fig. 18(a) and stripes in Fig. 18(b)), irregular patterns (Fig. 18(c) & (d)), fine-scale pattern (Fig. 18(e)), large-scale pattern (Fig. 18(f)), and solid black region (Fig. 18(g)).

Visual Comparison We first compare our method to a crisp-boundary extraction method proposed by Isola et al. [2014] in Fig. 13(a). As Isola's method is tailored for natural images, it generally fails to extract clear and smooth structural lines from manga images, especially when the structural lines are immersed in screen pattern. Furthermore, fine structural lines (e.g. hair and wrinkle on the gown) are missed since Isola's method mainly tailors for detecting large-scale object boundaries. In comparison, our method can extract clear and fine structural lines while removing screen pattern.

In Fig. 13(b), we compare our method to manga colorization [Qu et al. 2006], which is essentially a screen pattern segmentation method using Gabor wavelet analysis. Both methods can handle arbitrary screen patterns. As the texture analysis needs a window to analyze and the texture features near the boundary are substantially

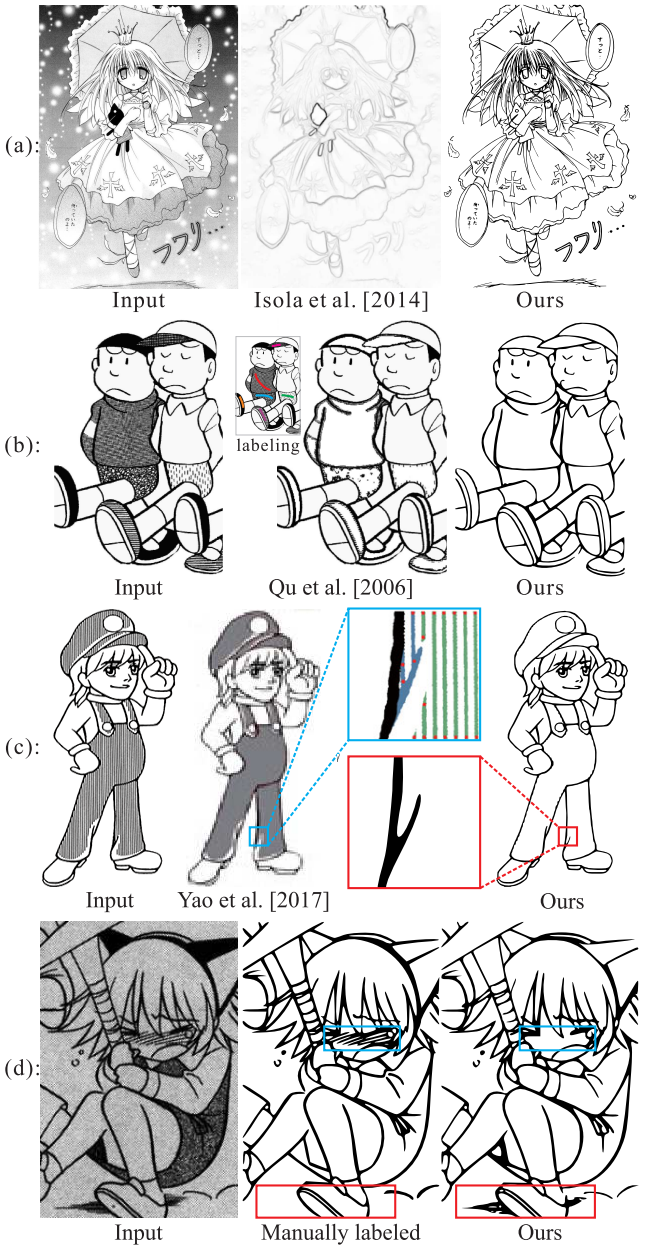


Fig. 13. Comparisons with Isola et al. [2014], Qu et al. [2006], Yao et al. [2017] and manually labeled result. ©Kiriga Yuki, Chih-Yuan Yao, Yagami Ken

different from that at the region center, noisy and unclear boundaries are usually obtained by their method, as demonstrated in the pants of the left boy. In contrast, our method can successfully obtain clear structural lines.

In Fig. 13(c), we compare our method to that of Yao et al. [2017]. If the screen pattern falls in one of the dot, grid or stripe categories (prior models), their method tries to model the pattern, as demonstrated in the blown-up blue box. Any misalignment or breakage of the stripes may interfere such modeling. On the other hand, since we do not explicitly model the patterns nor assume the type of pattern, our method is resistant to misalignment or breakage. Nevertheless, our method is still able to obtain comparable high-quality result. Yao et al. [2017] also proposed a method, based on relative total variation (RTV), to handle screen patterns besides the three prior models. Since we cannot obtain the original code of Yao et al. [2017], we compare to the RTV in the second column of Fig. 18 instead. Yao's method should share the similar problems as that of RTV, e.g. over-smoothing of pattern-immersed whiskers and collar of the cat (Fig. 18(d)) and inability to smooth large-scale screen pattern (Fig. 18(e)). In comparison, our method can faithfully extract the structural lines immersed in patterns of similar scale or even larger scale (the rightmost column of Fig. 18).

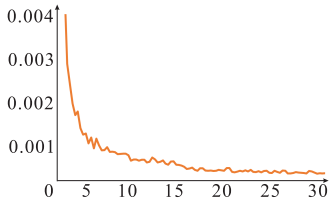
In Fig. 18, we compare our approach to more methods, including a texture smoothing method [Xu et al. 2012] (second column), a filter-based manga line and screen separation method [Ito et al. 2015] (third column), and an existing CNN model tailored for simplifying line drawings [Simo-Serra et al. 2016] (fourth column). The method of Ito et al. [2015] is the state-of-the-art automatic method for extracting structural lines from arbitrary screen patterns. The method of Xu et al. [2012] is originally proposed for image smoothing by utilizing the local oscillation assumption. Yao et al. [2017] adopted a similar metric for their application. Simo-Serra et al. [2016] proposed a CNN model to simplify sketches to clean line drawing. Note that the methods of Xu et al. [2012] and Simo-Serra et al. [2016] are not originally designed for structural line extraction. However, they are related to our goal and hence chosen for comparison.

In the second column of Fig. 18, we only show the smoothing result of Xu et al. [2012], without introducing further error in the line extraction. Their method cannot remove large-scale screen patterns (Fig. 18(a)&(f)) as such patterns violate the local oscillation assumption. It may also over-smooth the structural lines immersed in patterns (hair in Fig. 18(c) and collars in Fig. 18(d)). Ito et al. [2015] employed Laplacian of Gaussian filters and flow-based differences of Gaussian filters to remove screen patterns (the third column of Fig. 18). Their method shares similar difficulties in removing large-scale screen patterns (Fig. 18(f)) and irregular patterns (backgrounds of Fig. 18(c)&(d)), due to its filtering nature. For lines overlapped with the patterns (e.g. suit in dotted pattern in Fig. 18(a)), their extracted lines are also rough. The method of Simo-Serra [2016] (fourth column of Fig. 18) can remove most of the screen patterns, except the large-scale one (Fig. 18(d)). As their original application is sketch simplification, their method unavoidably groups the structural lines with the detailed structures in the patterns, and is sometimes overly aggressive in removing details including fine structural lines ((Fig. 18(b)-(g)). In sharp contrast, our method outperforms all competitors in terms of the abilities in removing screen patterns (regular, irregular or pictorial, fine-scale or large-scale) and preserving structural lines, even these lines are fully immersed into the screen patterns (whiskers and collar in Fig. 18) or overlapped with large-scale patterns (suit in Fig. 18(a)).

We also compare our result with manually labeled result (groundtruth) in Fig. 13(d). While our method produces visually similar results in most cases, these results may still differ from the groundtruth. For example, our method regards the hatching lines surrounding the nose as screen pattern and removes them completely, while

the human artist regards these lines are semantically important (expressing emotion) and preserves them (the blue box). Conversely, our method identifies the shadow lines behind the shoe as structures, but the artist knows that they do not depict the object structure and removes them (the red box). The differences are due to the fact that our model lacks of higher level semantic understanding of the visual content (e.g. emotion-expressing hatching and shadow-depicting pattern).

Quantitative Evaluation
To evaluate our method quantitatively, we first measure the training loss of our CNN model. It is the error between the convolved output from the network (using the current trained weights) and the ground truth in the training data. This loss is measured in terms of mean squared error (MSE) normalized by the image resolution to the range of [0, 1]. The right figure shows the training loss throughout the whole training epochs. It rapidly reduces after the first few epochs and stabilized at around 25 epochs.



Besides, we also evaluate the accuracy of our method on manga images that do not exist in our training data. They include both the synthetic and the real-world manga images. The synthetic ones are created from 39 line drawings that have not been used in the training data. A set of 137 manga images are created from these 39 line drawings using the generation method described in Section 4.1. These 39 line drawings are therefore the ground-truth for validation. Besides the synthetic ones, we also prepare 37 real-world manga images randomly selected from Manga109 database [Matsui et al. 2015]. The ground-truth structural line images are manually labeled by artists (e.g. Fig. 13(d)). More manually labeled data can be found in the supplementary materials.

In this quantitative evaluation, we compare our method to several state-of-the-art methods [Isola et al. 2014; Ito et al. 2015; Simo-Serra et al. 2016]. Note that the method proposed by Simo-Serra et al. [2016] is not originally trained for structural line extraction. For a fair comparison, we also train their model with our training data in order to better measure the improvement of our network design. During the experiment, we feed the testing manga images to our method and our competitors, and then measure the difference between the ground truth and results from different methods, in terms of peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and the region coverage rate (RCR). Here, RCR $= (E \wedge G)/(E \vee G)$, where E is the extracted image and G is the corresponding ground truth. Since binarization is needed for measuring RCR, we adopt the same binarization threshold $T_b = 225$ for all methods whenever necessary. The statistics for synthetic and real-world manga images in Tables 2 and 3 respectively.

For all three measurements metrics, the higher the values are, the more accurate the extraction results are. Our method achieves the highest accuracy in all three measurement metrics, for both synthetic and real-world manga images. The superiority is substantial. The method of Isola et al. [2014] is tailored for boundary detection in natural images, which only works well in detecting large and high-contrast structure. The method of Ito et al. [2015] relies on LoG and FDoG, which only works on a limited scale of the spatial domain and can only work well on regular and near-regular screen patterns. Their line abstraction method may also generate noisy lines. The method of Simo-Serra et al. [2016] achieves better PSNR due to the clarity of their output. However, its line abstraction and simplification goals may change the position and topology of the structural lines, and hence hurt its SSIM and RCR scores. With our training dataset, acceptable results can already be obtained with the plain network proposed by Simo-Serra et al. [2016]. However, as we have discussed in Section 4.2, the plain network is unable to produce high-quality results when the input content is rich and complex (e.g. Fig. 12). In contrast, our network model retains the position and topology of the structural lines with minimized roughness and noisiness.

	PSNR	SSIM	RCR
Isola et al. [2014]	14.151	0.7666	0.0967
Ito et al. [2015]	9.101	0.6328	0.3869
Simo-Serra et al. [2016]	12.852	0.5989	0.2884
Simo-Serra et al. [2016] with our training dataset	21.988	0.9268	0.6000
Ours	**28.050**	**0.9957**	**0.9444**

Table 2. Extraction accuracy on the synthetic manga images.

	PSNR	SSIM	RCR
Isola et al. [2014]	12.590	0.4908	0.1792
Ito et al. [2015]	16.383	0.8877	0.7631
Simo-Serra et al. [2016]	11.304	0.6150	0.2883
Simo-Serra et al. [2016] with our training dataset	16.974	0.8473	0.5701
Ours	**21.087**	**0.9502**	**0.8196**

Table 3. Extraction accuracy on real-world manga images.

Timing statistics All experiments are conducted on a PC with Intel Core i7-4710 2.5GHz, 16GB RAM. Our current system is implemented with the keras framework [Chollet 2015] accelerated by NVIDIA GeForce TITAN X (Maxwell) GPU. The offline training process takes approximately two days with 30 epochs of training. On average, the online filtering process consumes only 0.84 seconds in average to process 1 million pixels.

Limitations While our model can process manga images with arbitrary screen patterns, it still lacks very high-level semantic understanding to correctly handle sparse hand-drawn hatching. Occasionally, such hatching can be decorative and undesirable to remove (e.g. the blue box in Fig. 13(d)). In other cases, such hatching may only serve for shading purpose and can be removed (e.g. the red box in Fig. 13(d)). It is actually dependent on the user application need. Also, our method may get confused by small black region and thick structural line. For example, the upper eyelids of the girl in Fig. 14 are perceived as very thick structural lines to readers, but our method mistakenly identifies them as small black regions in the


(a) Input (b) Grayscale result (c) Vectorized result

Fig. 14. Limitations. (a) Input. (b) Our method mistakenly identifies the upper eyelids of the girl (very thick structural lines) as small black regions. It also fails to recognize the step-edges around the reflection of the eyeballs. (c) The binarization may lead to occasional loss of structural lines, such as the discontinuous hairs. ©Kiriga Yuki

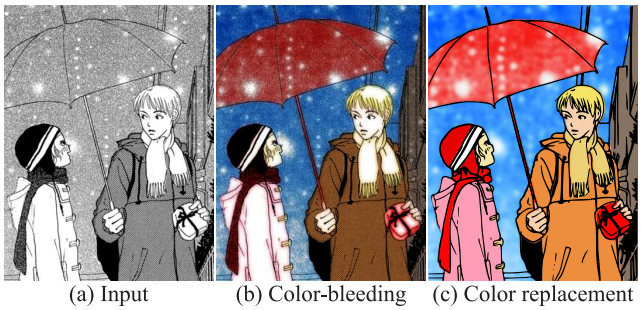

(a) Input (b) Color-bleeding (c) Color replacement

Fig. 15. An example of manga colorization. ©Yamada Uduki

result. Besides, since our method is tailored for extracting structural lines instead of step edges, it may fail to recognize the step edges (e.g. the reflection of the eyeballs in Fig. 14). Additionally, our current binarization strategy in vectorization is quite naive. It does not adapt to the local statistics of grayness, resulting in occasional loss of structural lines (e.g. hairs in Fig. 14). The thickness of the structural lines after binarization and vectorization is also not guaranteed to be the same with the input. Furthermore, since learning-based method strongly relies on the training data. Bias could exist if the variety of training data is insufficient, or the training data is not carefully prepared and balanced for diversity of styles. Therefore, if the training data is too dissimilar from the actual manga, the result could be unsatisfactory.

6 APPLICATIONS

With the clean structual lines extracted, we can utilize the result in many interesting digital manga applications. Here, we demonstrate three potential applications, including including manga colorization, manga retargeting, and 2.5D manga generation.

Manga Colorization As legacy mangas are mostly in black-and-white, manga colorization is probably the first interesting extension for digital reproduction of the mangas. With the extracted high-quality structural lines, we can easily and accurately separate the screened regions for colorization. Fig. 15 presents two styles of colorization. Fig. 15(b) demonstrates the color-bleeding style [Qu et al. 2006] in which the screen pattern is retained. Fig. 15(c) demonstrates the style of color replacement, in which the screen pattern is replaced by a colorized region with the spatially varying intensity that ressembles the original spatially varying screen tone. Comparing to the texture-analysis based manga colorization [Qu et al. 2006], we can reproduce clearer and smoother boundaries.

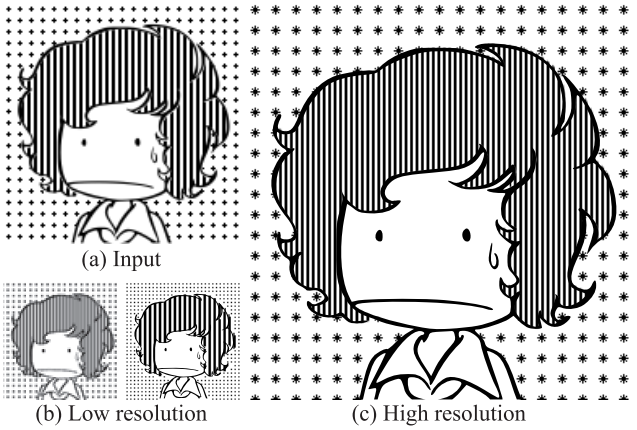

(a) Input (b) Low resolution (c) High resolution

Fig. 16. An example of manga retargeting. ©Minamoto Tarou

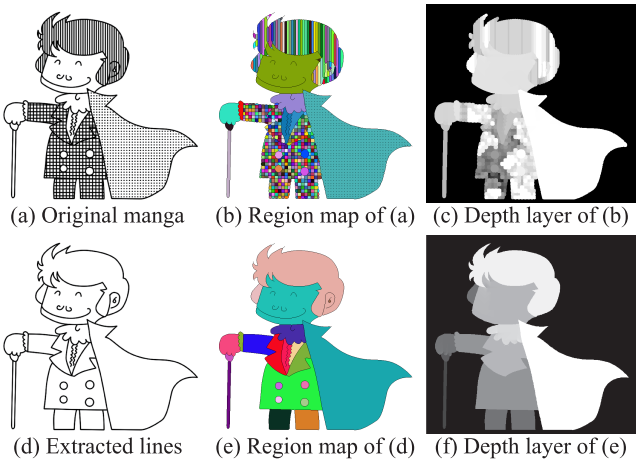

(a) Original manga (b) Region map of (a) (c) Depth layer of (b)
(d) Extracted lines (e) Region map of (d) (f) Depth layer of (e)

Fig. 17. An example of 2.5D manga generation. ©Minamoto Tarou

Manga Retargeting With varying display resolutions of modern portable devices, naive resampling may lead to blurry presentation of lines and screen patterns on low-resolution displays (left of Fig. 16(b)), or annoying blocky appearance on high-resolution displays. Once the structural lines and screen patterns are separated, we can retarget the manga to different display resolutions with higher visual quality. We first vectorize the structural lines, so that we rasterize clear structural lines in the target resolution. For screen patterns, we can adopt texture synthesis [Barnes et al. 2009] to obtain a larger pattern. Note that it may not be desirable to proportionally scale up or down the screen pattern to the target resolution, because each screen pattern should have a valid range of display resolution. For instance, proportionally scaling up the stripe pattern (hair in Fig. 16(a)) may loose its original tonal meaning and turns it into structure. In Fig. 16(b) and (c), we impose a lower and upper bound on the scaling of impainted stripe pattern to obtain sensible visual appearance. We can even introduce the level-of-details to present the screen pattern smartly. For instance, the cross pattern in the background region of the original manga

can be replaced by a dot pattern for low-resolution display (right of Fig. 16(b)), and replaced by a star-shaped pattern for high-resolution display (Fig. 16(c)). This is analogous to hinting in typography.

2.5D Manga Generation With the clear structural lines, we can also deduce the depth semantics to produce 2.5D manga. Fig. 17 shows one such example. To do so, we first obtain the region map (Fig. 17(e)) from our clear structural line image (Fig. 17(d)). Then we adopt the T-junction analysis [Liu et al. 2013] to deduce the depth ordering of these regions (Fig. 17(f)). With the 2.5D manga, we can displace our viewpoint, or even generate a stereoscopic manga pair. On the other hand, depth analysis directly on the original manga is not feasible due to the interference of highly-structured screen patterns. Excessive amount of regions may be extracted (Fig. 17(b)), leading to a lot of errors in the followed depth analysis (Fig. 17(c)).

7 CONCLUSION

In this paper, we proposed a CNN-based method for extracting clear and smooth structural lines from screen-rich mangas. A tailor-made CNN model of deeper network structure is proposed to handle the large variety of screen patterns and increase output accuracy. We also developed a practical way to generate a rich training data set. Our system significantly outperforms the existing methods in terms of visual quality and quantitative measurements, without making assumption on the type of screen patterns. We also demonstrate multiple potential applications utilizing our high-quality extraction results.

Hand-drawn contents (such as hatching and fine details) are sometimes hard to be classified as screen pattern or structural details. The interpretation is also application-dependent. Currently, the removal of screen patterns is fully automatic. Therefore, the future direction is to have an intelligent GUI that allows user to selectively enable/disable the removal of the screen patterns, especially those ambiguous hand-drawn patterns. Such user guidance or feedback can be further used to train the CNN model to adapt to a specific user need.

REFERENCES

Pablo Arbelaez, Michael Maire, Charless C. Fowlkes, and Jitendra Malik. 2011. Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 5 (2011), 898–916.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28, 3 (Aug. 2009).

John Canny. 1986. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8, 6 (1986), 679–698.

François Chollet. 2015. Keras. https://github.com/fchollet/keras. (2015).

James Hays, Marius Leordeanu, Alexei A Efros, and Yanxi Liu. 2006. Discovering texture regularity as a higher-order correspondence problem. In *Computer Vision–ECCV 2006*. Springer, 522–535.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity Mappings in Deep Residual Networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*. 630–645.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 448–456.

Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. 2014. Crisp Boundary Detection Using Pointwise Mutual Information. In *ECCV*.

Kota Ito, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2015. Separation of Manga Line Drawings and Screentones. In *Eurographics 2015 - Short Papers, Zurich, Switzerland, May 4-8, 2015*. 73–76.

Henry Kang, Seungyong Lee, and Charles K. Chui. 2007. Coherent line drawing. In *Proceedings of the 5th International Symposium on Non-Photorealistic Animation and Rendering 2007, San Diego, California, USA, August 4-5, 2007*. 43–50.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

Iasonas Kokkinos. 2015. Pushing the boundaries of boundary detection using deep learning. *arXiv preprint arXiv:1511.07386* (2015).

Johannes Kopf and Dani Lischinski. 2012. Digital reconstruction of halftoned color comics. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 140.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. 1106–1114.

Xueting Liu, Xiangyu Mao, Xuan Yang, Linling Zhang, and Tien-Tsin Wong. 2013. Stereoscopizing Cel Animations. *ACM Transactions on Graphics (SIGGRAPH Asia 2013 issue)* 32, 6 (November 2013), 223:1–223:10.

Yanxi Liu, Tamara Belkina, James Hays, and Roberto Lublinerman. 2008. Image Defencing. In *Proceedings of CVPR 2008*.

Yanxi Liu, Robert T Collins, and Yanghai Tsin. 2004a. A computational model for periodic pattern perception based on frieze and wallpaper groups. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26, 3 (2004), 354–371.

Yanxi Liu, Wen-Chieh Lin, and James Hays. 2004b. Near-regular texture analysis and manipulation. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 368–376.

Xiangyu Mao, Xueting Liu, Tien-Tsin Wong, and Xuemiao Xu. 2015. Region-based structure line detection for cartoons. *Computational Visual Media* 1, 1 (2015), 69–78.

Yusuke Matsui, Kota Ito, Yuji Aramaki, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2015. Sketch-based Manga Retrieval using Manga109 Dataset. *CoRR* abs/1510.04389 (2015).

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. 2015. Learning Deconvolution Network for Semantic Segmentation. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 1520–1528.

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

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*. 234–241. DOI:https://doi.org/10.1007/978-3-319-24574-4_28

Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. 2015. DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 3982–3991.

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Trans. Graph.* 35, 4 (2016), 121.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556 (2014).

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2014. Striving for Simplicity: The All Convolutional Net. *CoRR* abs/1412.6806 (2014).

Daniel Sýkora, Jan Buriánek, and Jiří Žára. 2004. Unsupervised colorization of black-and-white cartoons. In *International Symposium on Non-Photorealistic Animation and Rendering*. 121–127.

Carlo Tomasi and Roberto Manduchi. 1998. Bilateral Filtering for Gray and Color Images. In *ICCV*. 839–846.

Luminita A. Vese and Stanley Osher. 2003. Modeling Textures with Total Variation Minimization and Oscillating Patterns in Image Processing. *J. Sci. Comput.* 19, 1-3 (2003), 553–572.

Saining Xie and Zhuowen Tu. 2015. Holistically-Nested Edge Detection. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. 1395–1403.

Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. 2011. Image smoothing via L_0 gradient minimization. *ACM Trans. Graph.* 30, 6 (2011), 174.

Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. 2012. Structure extraction from texture via relative total variation. *ACM Trans. Graph.* 31, 6 (2012), 139.

Chih-Yuan Yao, Shih-Hsuan Hung, Guo-Wei Li, I-Yu Chen, Reza Adhitya, and Yu-Chi Lai. 2017. Manga Vectorization and Manipulation with Procedural Simple Screentone. *IEEE Trans. Vis. Comput. Graph.* 23, 2 (2017), 1070–1084.

Song-Hai Zhang, Tao Chen, Yi-Fei Zhang, Shi-Min Hu, and Ralph R Martin. 2009. Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics* 15, 4 (2009), 618–629.

Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. 2016. Loss Functions for Neural Networks for Image Processing. *CoRR* abs/1511.08861 (2016).

Fig. 18. Comparisons with RTV [Xu et al. 2012], Ito et al. [2015], and Simo-Serra et al. [2016]. ©Minamoto Tarou, Okuda Momoko, Kiriga Yuki and Ebihurai.