Generating Manga from Illustrations via Mimicking Manga Creation Workflow

Style2Paints Research 2021 Presents

Supplementary Material

We provide additional results, comparisons, and implementation details of our framework and experiments.

Input illustration | Greyscale | Mimicking manga workflow (Ours)

Contents

1 Additional Results **2**

1.1 Qualitative results . . . 2

1.2 Additional comparisons . . . 63

2 Dataset Details **77**

2.1 Example annotations . . . 77

3 Implementation Details **79**

3.1 Neural network architecture . . . 79

3.2 Implementation details of previous methods . . . 79

3.2.1 Clip Studio Paints [2020] . . . 79

3.2.2 Qu et al. [2008] . . . 80

3.2.3 Isola et al. [2017] (Pix2Pix) . . . 80

3.2.4 Li et al. [2019] . . . 80

3.2.5 Xie et al. [2020] . . . 80

4 Copyrights and Ethics Statements **80**

Figure 1: **Quantitative result 1.** On the left is the original input, and the result is on the right.

1 Additional Results

1.1 Qualitative results

We present 60 additional qualitative results as shown in Fig. 1-60. All those 60 input images are unseen from our dataset.

Figure 2: **Quantitative result 2.** On the left is the original input, and the result is on the right.

Figure 3: **Quantitative result 3.** On the left is the original input, and the result is on the right.

Figure 4: **Quantitative result 4.** On the left is the original input, and the result is on the right.

Figure 5: **Quantitative result 5.** On the left is the original input, and the result is on the right.

Figure 6: **Quantitative result 6.** On the left is the original input, and the result is on the right.

Figure 7: **Quantitative result 7.** On the left is the original input, and the result is on the right.

Figure 8: **Quantitative result 8.** On the left is the original input, and the result is on the right.

Figure 9: **Quantitative result 9.** On the left is the original input, and the result is on the right.

Figure 10: **Quantitative result 10.** On the left is the original input, and the result is on the right.

Figure 11: **Quantitative result 11.** On the left is the original input, and the result is on the right.

Figure 12: **Quantitative result 12.** On the left is the original input, and the result is on the right.

Figure 13: **Quantitative result 13.** On the left is the original input, and the result is on the right.

Figure 14: **Quantitative result 14.** On the left is the original input, and the result is on the right.

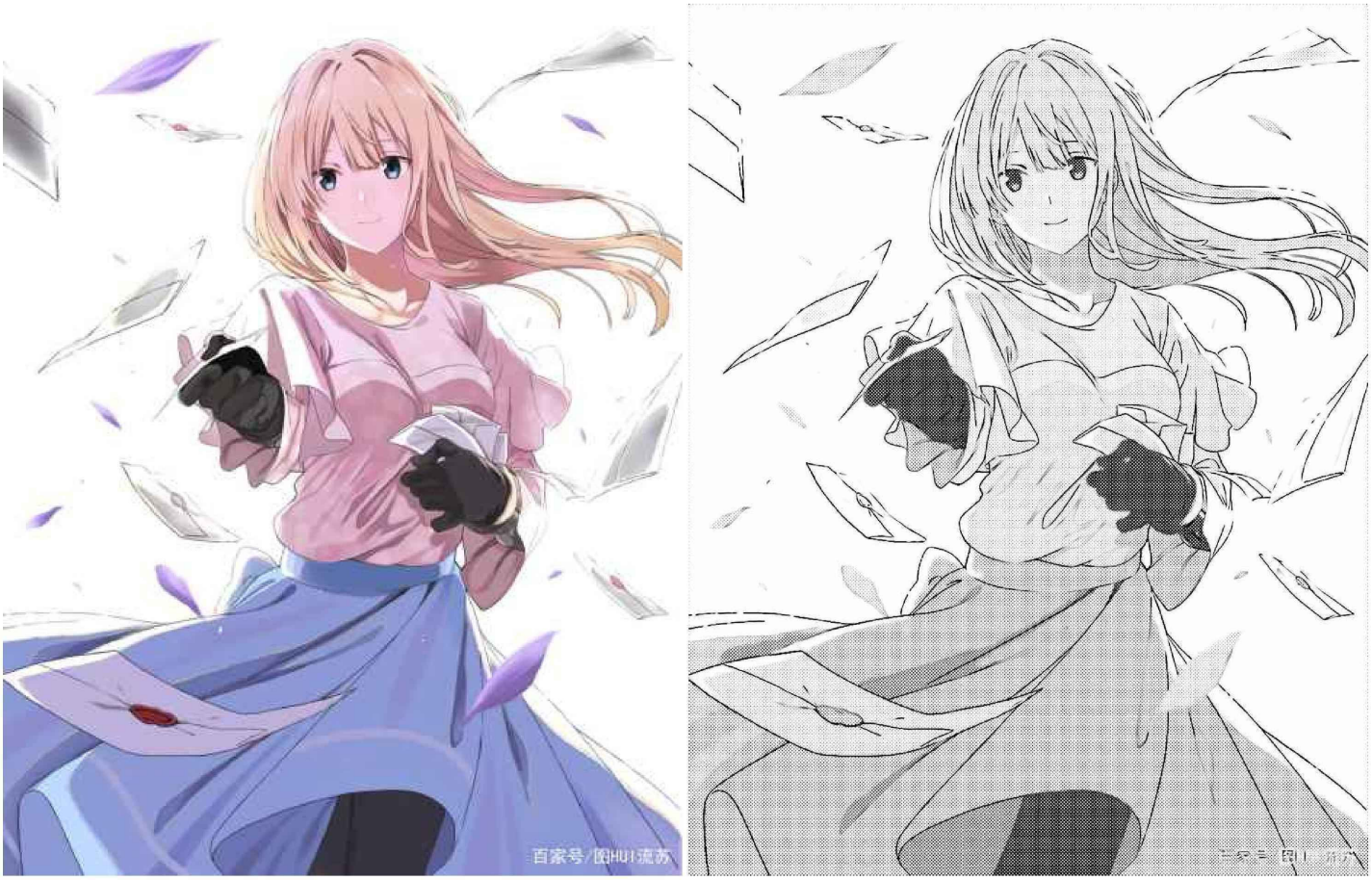


Figure 15: **Quantitative result 15.** On the left is the original input, and the result is on the right.

Figure 16: **Quantitative result 16.** On the left is the original input, and the result is on the right.

Figure 17: **Quantitative result 17.** On the left is the original input, and the result is on the right.

Figure 18: **Quantitative result 18.** On the left is the original input, and the result is on the right.

Figure 19: **Quantitative result 19.** On the left is the original input, and the result is on the right.

Figure 20: **Quantitative result 20.** On the left is the original input, and the result is on the right.

Figure 21: **Quantitative result 21.** On the left is the original input, and the result is on the right.

Figure 22: **Quantitative result 22.** On the left is the original input, and the result is on the right.

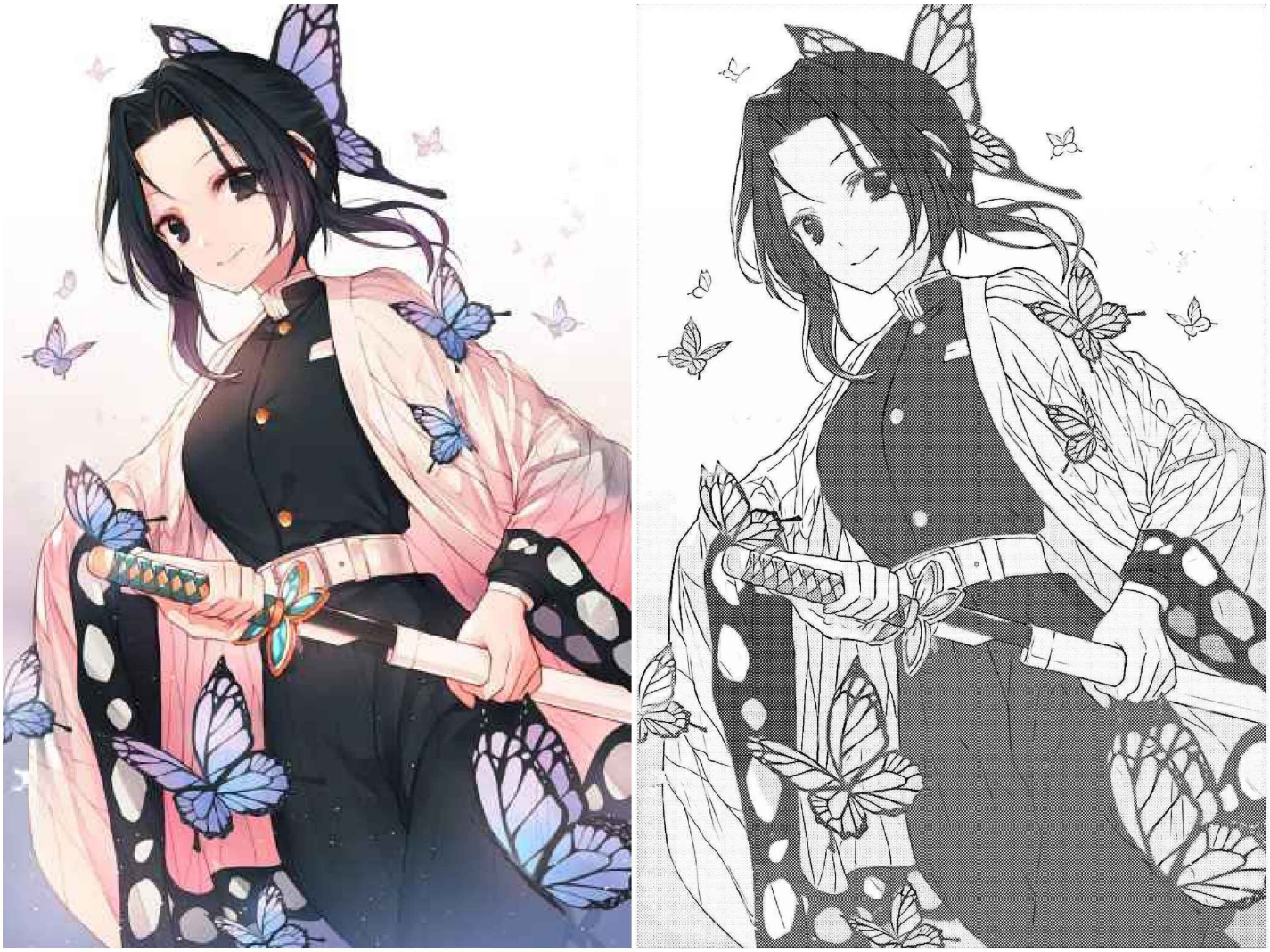


Figure 23: **Quantitative result 23.** On the left is the original input, and the result is on the right.

Figure 24: **Quantitative result 24.** On the left is the original input, and the result is on the right.

Figure 25: **Quantitative result 25.** On the left is the original input, and the result is on the right.

Figure 26: **Quantitative result 26.** On the left is the original input, and the result is on the right.

Figure 27: **Quantitative result 27.** On the left is the original input, and the result is on the right.

Figure 28: **Quantitative result 28.** On the left is the original input, and the result is on the right.

Figure 29: **Quantitative result 29.** On the left is the original input, and the result is on the right.

Figure 30: **Quantitative result 30.** On the left is the original input, and the result is on the right.

Figure 31: **Quantitative result 31.** On the left is the original input, and the result is on the right.

Figure 32: **Quantitative result 32.** On the left is the original input, and the result is on the right.

Figure 33: **Quantitative result 33.** On the left is the original input, and the result is on the right.

Figure 34: **Quantitative result 34.** On the left is the original input, and the result is on the right.

Figure 35: **Quantitative result 35.** On the left is the original input, and the result is on the right.

Figure 36: **Quantitative result 36.** On the left is the original input, and the result is on the right.

Figure 37: **Quantitative result 37.** On the left is the original input, and the result is on the right.

Figure 38: **Quantitative result 38.** On the left is the original input, and the result is on the right.

Figure 39: **Quantitative result 39.** On the left is the original input, and the result is on the right.

Figure 40: **Quantitative result 40.** On the left is the original input, and the result is on the right.

Figure 41: **Quantitative result 41.** On the left is the original input, and the result is on the right.

Figure 42: **Quantitative result 42.** On the left is the original input, and the result is on the right.

Figure 43: **Quantitative result 43.** On the left is the original input, and the result is on the right.

Figure 44: **Quantitative result 44.** On the left is the original input, and the result is on the right.

Figure 45: **Quantitative result 45.** On the left is the original input, and the result is on the right.

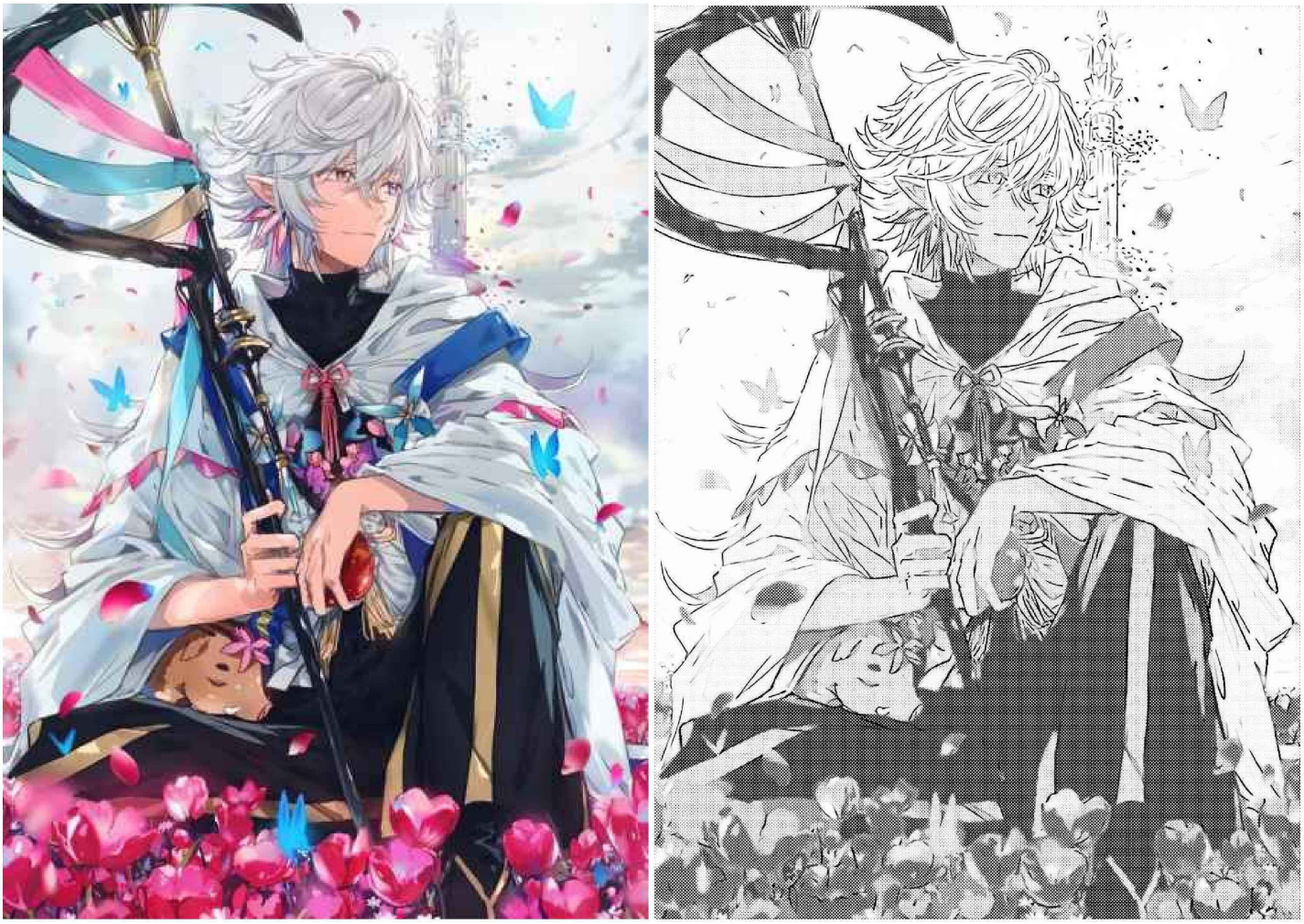


Figure 46: **Quantitative result 46.** On the left is the original input, and the result is on the right.

Figure 47: **Quantitative result 47.** On the left is the original input, and the result is on the right.

Figure 48: **Quantitative result 48.** On the left is the original input, and the result is on the right.

Figure 49: **Quantitative result 49.** On the left is the original input, and the result is on the right.

Figure 50: **Quantitative result 50.** On the left is the original input, and the result is on the right.

Figure 51: **Quantitative result 51.** On the left is the original input, and the result is on the right.

Figure 52: **Quantitative result 52.** On the left is the original input, and the result is on the right.

Figure 53: **Quantitative result 53.** On the left is the original input, and the result is on the right.

Figure 54: **Quantitative result 54.** On the left is the original input, and the result is on the right.

Figure 55: **Quantitative result 55.** On the left is the original input, and the result is on the right.

Figure 56: **Quantitative result 56.** On the left is the original input, and the result is on the right.

Figure 57: **Quantitative result 57.** On the left is the original input, and the result is on the right.

Figure 58: **Quantitative result 58.** On the left is the original input, and the result is on the right.

Figure 59: **Quantitative result 59.** On the left is the original input, and the result is on the right.

Figure 60: **Quantitative result 60.** On the left is the original input, and the result is on the right.

Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]

Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 61: **Additional Comparison 1.** We present additional comparisons between our framework and other alternative methods.

Figure 62: **Additional Comparison 2.** We present additional comparisons between our framework and other alternative methods.

1.2 Additional comparisons

We present 65 additional comparisons between our framework and possible alternative methods as shown in Fig. 61-75. All those 60 tested images are unseen from our dataset.

Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]

Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 63: **Additional Comparison 3.** We present additional comparisons between our framework and other alternative methods.

Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]

Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 64: **Additional Comparison 4.** We present additional comparisons between our framework and other alternative methods.

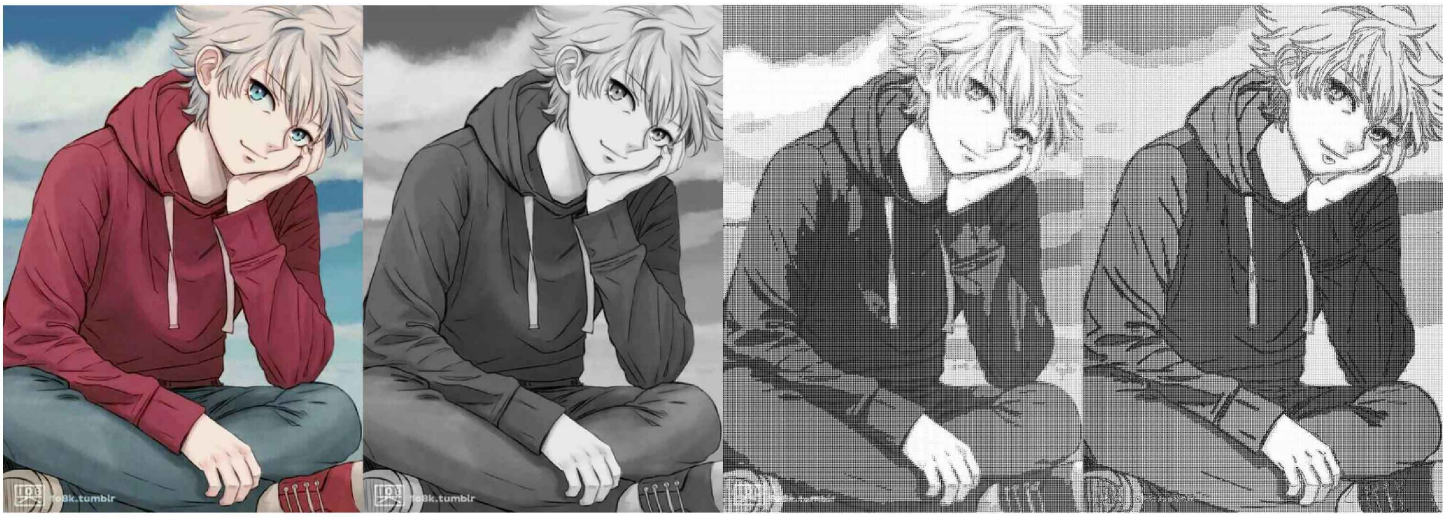
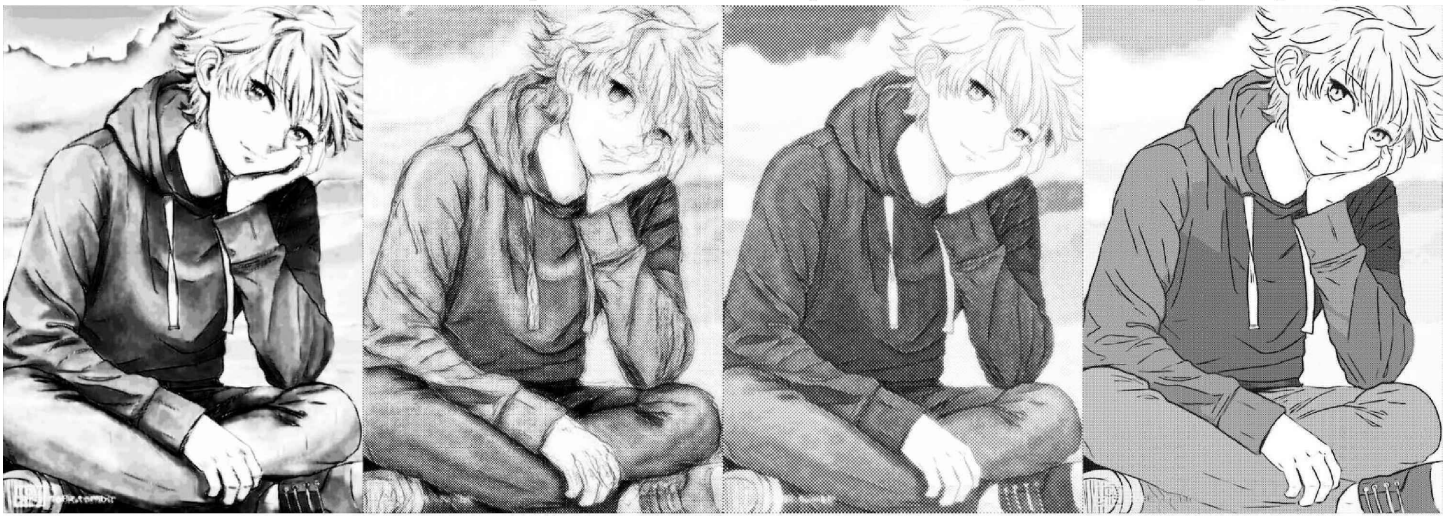


Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]



Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 65: **Additional Comparison 5.** We present additional comparisons between our framework and other alternative methods.

Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]

Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 66: **Additional Comparison 6.** We present additional comparisons between our framework and other alternative methods.

Figure 67: **Additional Comparison 7.** We present additional comparisons between our framework and other alternative methods.

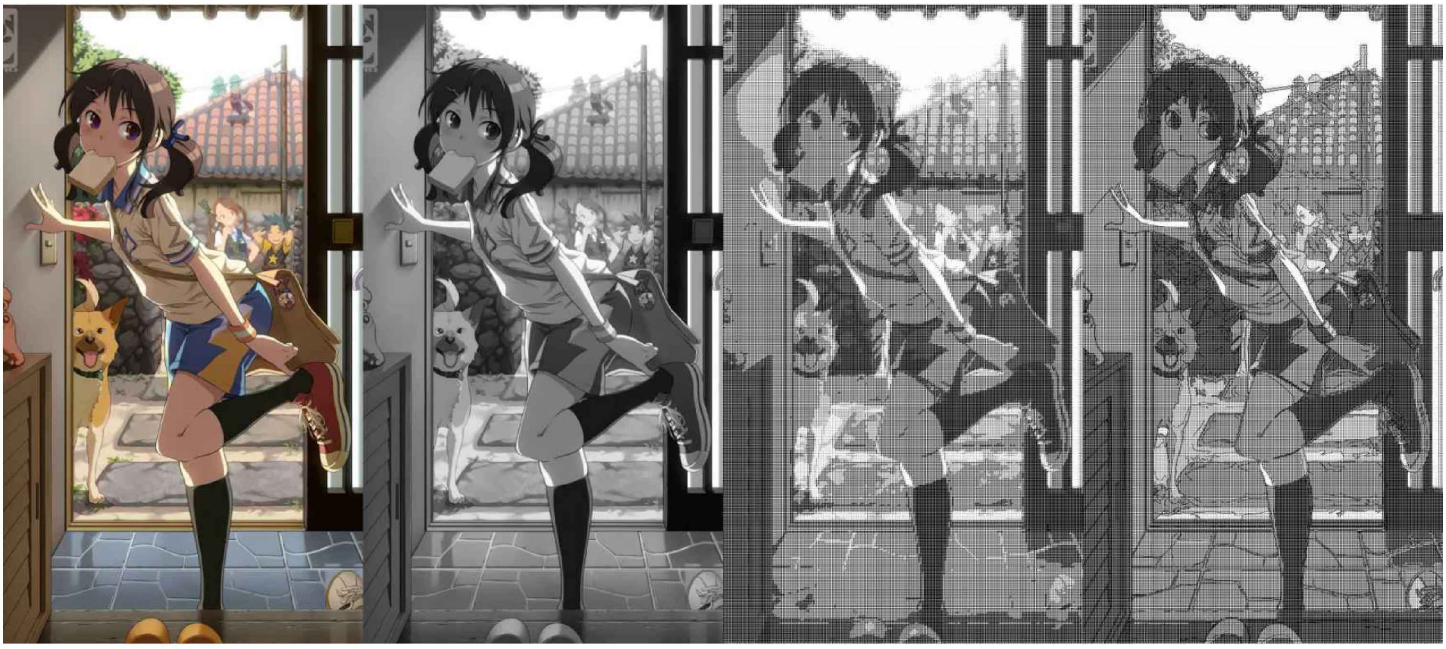
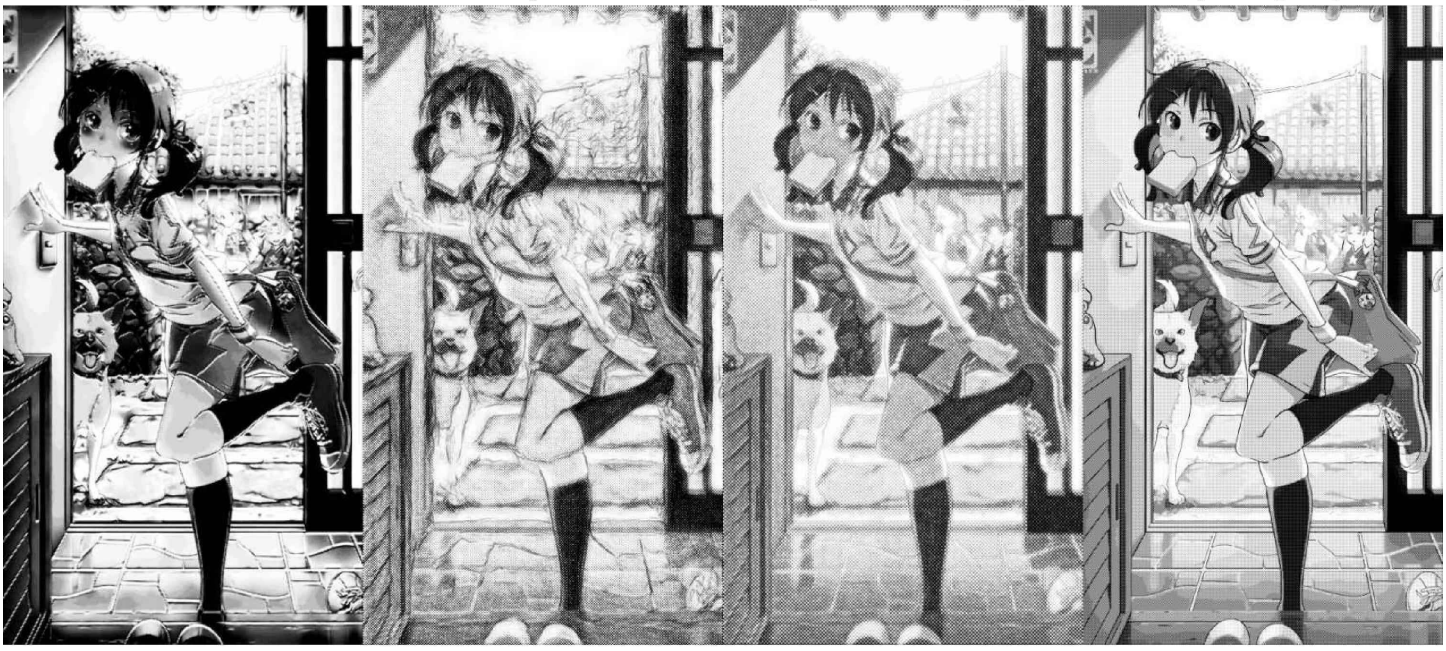


Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]



Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 68: **Additional Comparison 8.** We present additional comparisons between our framework and other alternative methods.

Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]

Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 69: **Additional Comparison 9.** We present additional comparisons between our framework and other alternative methods.

Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]

Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 70: **Additional Comparison 10.** We present additional comparisons between our framework and other alternative methods.

Figure 71: **Additional Comparison 11.** We present additional comparisons between our framework and other alternative methods.

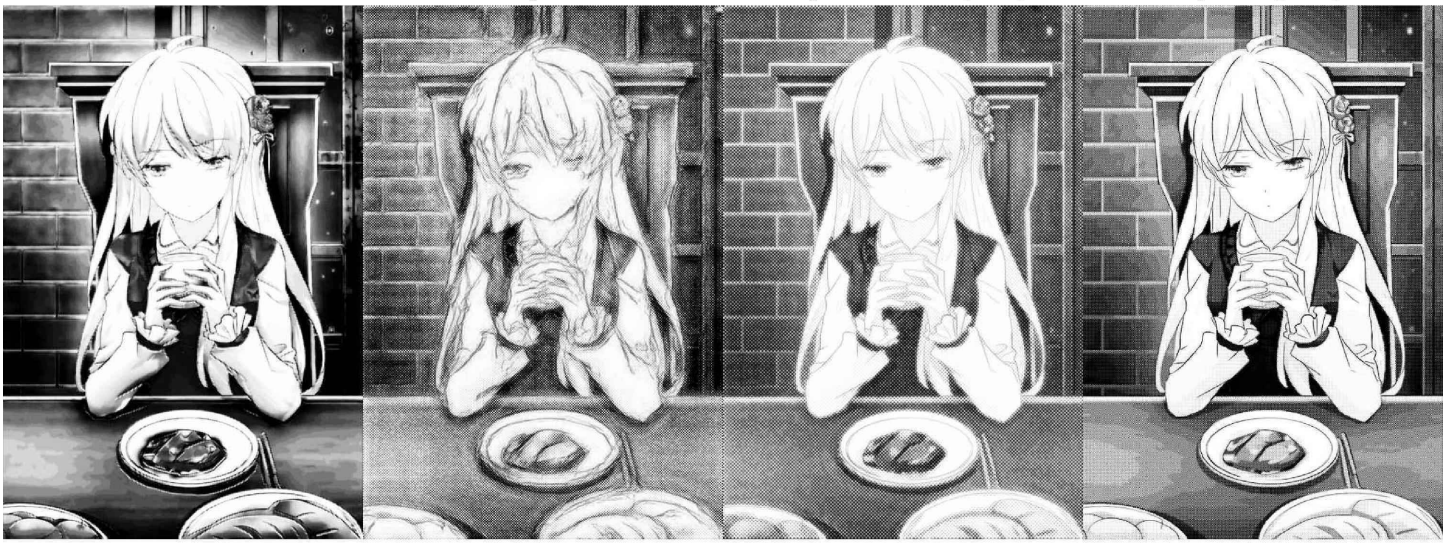
Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]

Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 72: **Additional Comparison 12.** We present additional comparisons between our framework and other alternative methods.

Figure 73: **Additional Comparison 13.** We present additional comparisons between our framework and other alternative methods.

Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]



Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 74: **Additional Comparison 14.** We present additional comparisons between our framework and other alternative methods.

Illustration | Greyscale | Clip Studio Paints [2020] | Qu et al. [2008]

Pix2pix, Isola et al. [2017] | Li et al. [2019] | Xie et al. [2020] | Ours

Figure 75: **Additional Comparison 15.** We present additional comparisons between our framework and other alternative methods.

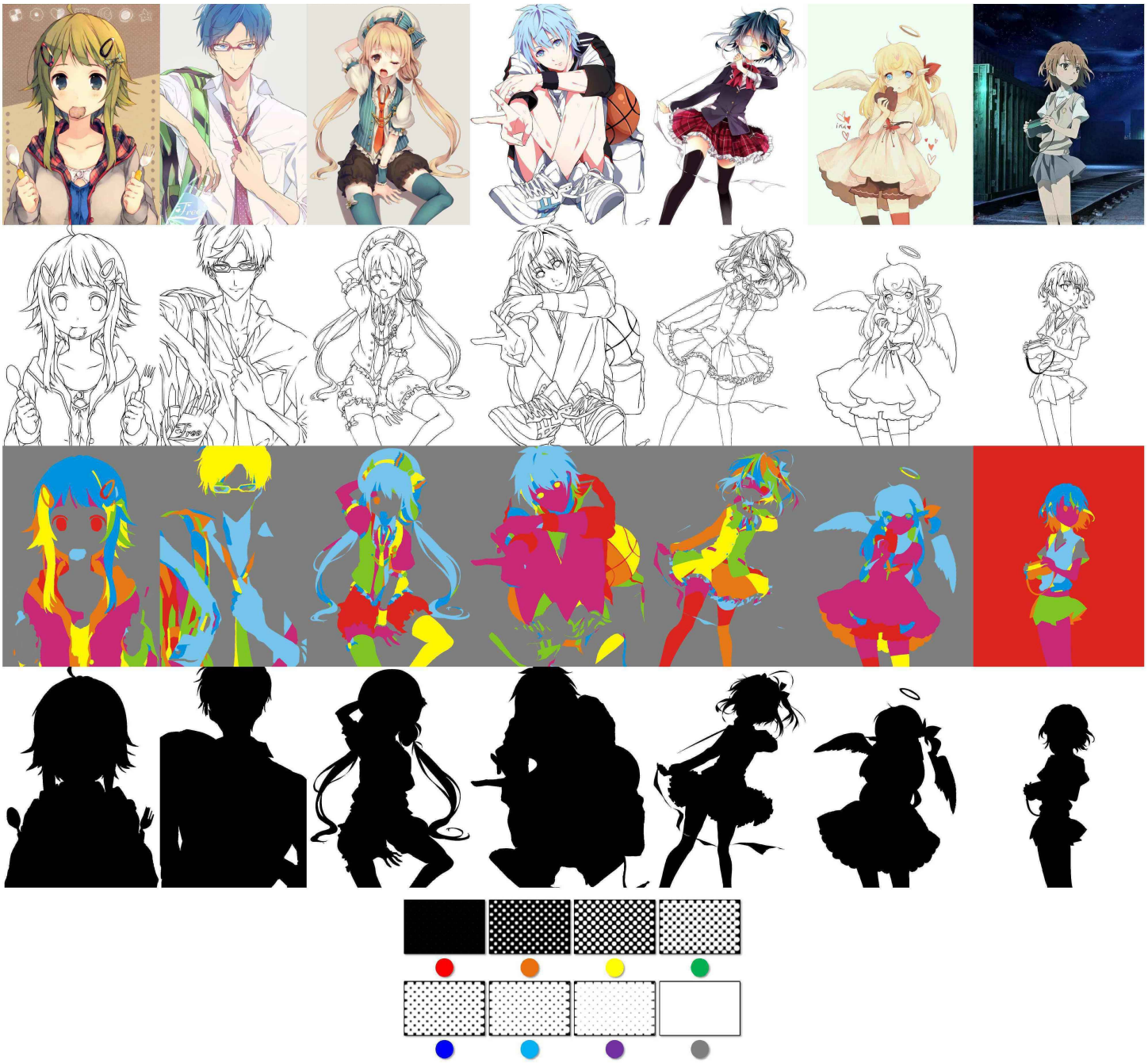


Figure 76: We present examples of annotations in our dataset. In the first row are the illustrations, the line drawings are in the second row, the regular screentone segmentations are in the third row, and the irregular screen texture masks are in the forth row. We also present a legend of the screentone sheets in the last row.

2 Dataset Details

2.1 Example annotations

We present examples of annotations in our dataset in Fig. 76 and Fig. 77.

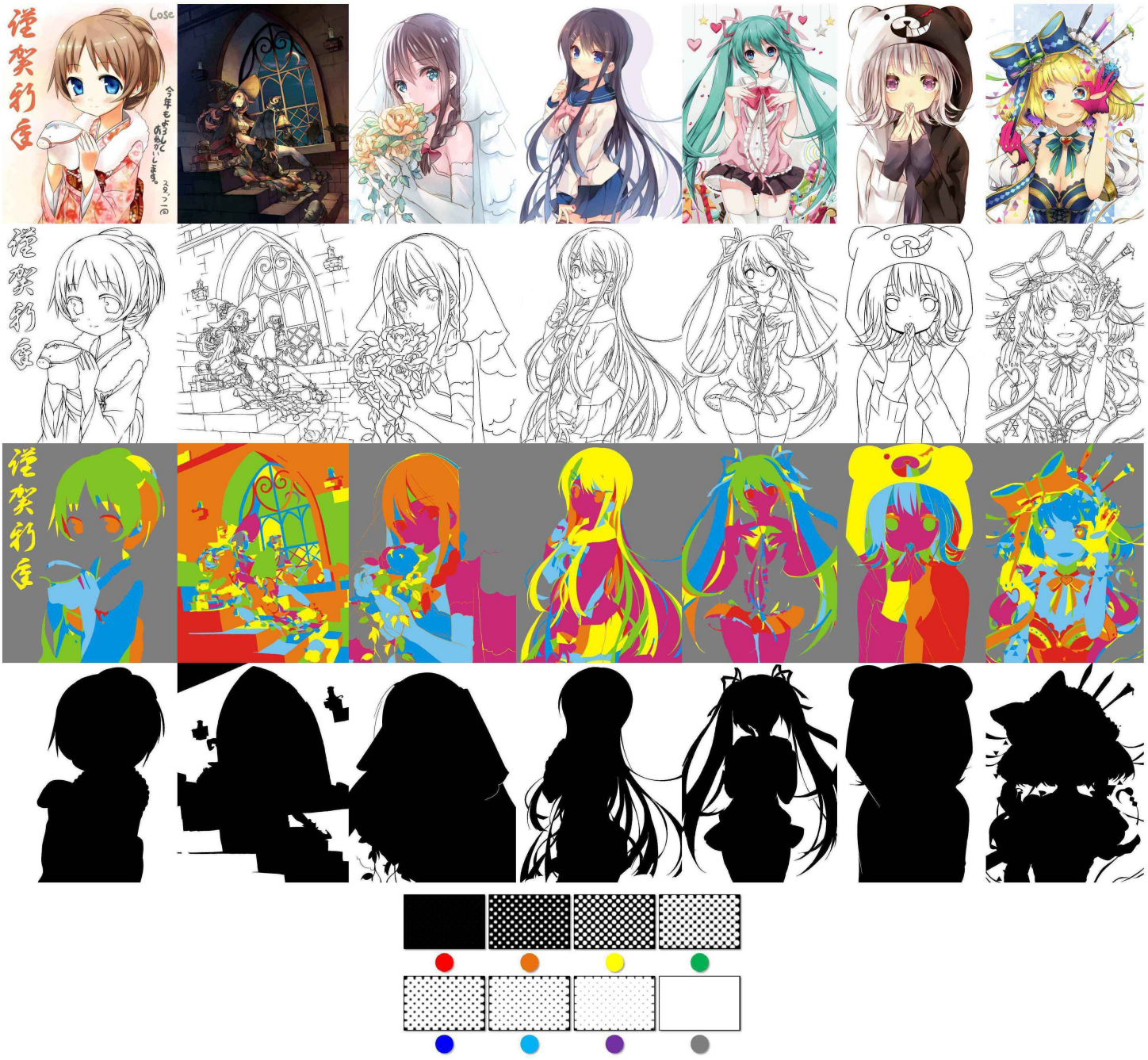


Figure 77: We present examples of annotations in our dataset. In the first row are the illustrations, the line drawings are in the second row, the regular screentone segmentations are in the third row, and the irregular screen texture masks are in the forth row. We also present a legend of the screentone sheets in the last row.

3 Implementation Details

3.1 Neural network architecture

We present a python (Keras style) implementation of the neural network architecture in our framework.

Code 1. A Keras implementation of the neural network architecture.

```
from keras.layers import Conv2D, Activation, Input, AveragePooling2D, UpSampling2D, Add, BatchNormalization
from keras.models import Model

input = Input(shape=(None, None, 3))
conv1a = Conv2D(filters=32, strides=(1, 1), kernel_size=(3, 3), padding='same')(input)
conv1a = BatchNormalization()(conv1a)
conv1a = Activation('relu')(conv1a)
conv1b = Conv2D(filters=32, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv1a)
conv1b = BatchNormalization()(conv1b)
conv1b = Activation('relu')(conv1b)
conv2a = AveragePooling2D(pool_size=(2, 2))(conv1b)
conv2a = Conv2D(filters=64, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv2a)
conv2a = BatchNormalization()(conv2a)
conv2a = Activation('relu')(conv2a)
conv2b = Conv2D(filters=64, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv2a)
conv2b = BatchNormalization()(conv2b)
conv2b = Activation('relu')(conv2b)
conv3a = AveragePooling2D(pool_size=(2, 2))(conv2b)
conv3a = Conv2D(filters=128, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv3a)
conv3a = BatchNormalization()(conv3a)
conv3a = Activation('relu')(conv3a)
conv3b = Conv2D(filters=128, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv3a)
conv3b = BatchNormalization()(conv3b)
conv3b = Activation('relu')(conv3b)
conv4a = AveragePooling2D(pool_size=(2, 2))(conv3b)
conv4a = Conv2D(filters=256, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv4a)
conv4a = BatchNormalization()(conv4a)
conv4a = Activation('relu')(conv4a)
conv4b = Conv2D(filters=512, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv4a)
conv4b = BatchNormalization()(conv4b)
conv4b = Activation('relu')(conv4b)
conv4c = Conv2D(filters=512, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv4b)
conv4c = BatchNormalization()(conv4c)
conv4c = Activation('relu')(conv4c)
conv4d = Conv2D(filters=256, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv4c)
conv4d = BatchNormalization()(conv4d)
conv4d = Activation('relu')(conv4d)
conv5a = UpSampling2D(size=(2, 2))(conv4d)
conv5a = Conv2D(filters=128, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv5a)
conv5a = Add()([conv5a, conv3b])
conv5a = BatchNormalization()(conv5a)
conv5a = Activation('relu')(conv5a)
conv5b = Conv2D(filters=128, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv5a)
conv5b = BatchNormalization()(conv5b)
conv5b = Activation('relu')(conv5b)
conv6a = UpSampling2D(size=(2, 2))(conv5b)
conv6a = Conv2D(filters=64, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv6a)
conv6a = Add()([conv6a, conv2b])
conv6a = BatchNormalization()(conv6a)
conv6a = Activation('relu')(conv6a)
conv6b = Conv2D(filters=64, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv6a)
conv6b = BatchNormalization()(conv6b)
conv6b = Activation('relu')(conv6b)
conv7a = UpSampling2D(size=(2, 2))(conv6b)
conv7a = Conv2D(filters=32, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv7a)
conv7a = Add()([conv7a, conv1b])
conv7a = BatchNormalization()(conv7a)
conv7a = Activation('relu')(conv7a)
conv7b = Conv2D(filters=32, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv7a)
conv7b = BatchNormalization()(conv7b)
conv7b = Activation('relu')(conv7b)
inking_decoder_output = Conv2D(filters=1, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv7b)
seg_conv4b = Conv2D(filters=512, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv4a)
seg_conv4b = Add()([seg_conv4b, conv4b])
seg_conv4b = BatchNormalization()(seg_conv4b)
seg_conv4b = Activation('relu')(seg_conv4b)
seg_conv4c = Conv2D(filters=512, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv4b)
seg_conv4c = Add()([seg_conv4c, conv4c])
seg_conv4c = BatchNormalization()(seg_conv4c)
seg_conv4c = Activation('relu')(seg_conv4c)
seg_conv4d = Conv2D(filters=256, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv4c)
seg_conv4d = Add()([seg_conv4d, conv4d])
seg_conv4d = BatchNormalization()(seg_conv4d)
seg_conv4d = Activation('relu')(seg_conv4d)
seg_conv5a = UpSampling2D(size=(2, 2))(seg_conv4d)
seg_conv5a = Conv2D(filters=128, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv5a)
seg_conv5a = Add()([seg_conv5a, conv5a])
seg_conv5a = BatchNormalization()(seg_conv5a)
seg_conv5a = Activation('relu')(seg_conv5a)
seg_conv5b = Conv2D(filters=128, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv5a)
seg_conv5b = Add()([seg_conv5b, conv5b])
seg_conv5b = BatchNormalization()(seg_conv5b)
seg_conv5b = Activation('relu')(seg_conv5b)
seg_conv6a = UpSampling2D(size=(2, 2))(seg_conv5b)
seg_conv6a = Conv2D(filters=64, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv6a)
seg_conv6a = Add()([seg_conv6a, conv6a])
seg_conv6a = BatchNormalization()(seg_conv6a)
seg_conv6a = Activation('relu')(seg_conv6a)
seg_conv6b = Conv2D(filters=64, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv6a)
seg_conv6b = Add()([seg_conv6b, conv6b])
seg_conv6b = BatchNormalization()(seg_conv6b)
seg_conv6b = Activation('relu')(seg_conv6b)
seg_conv7a = UpSampling2D(size=(2, 2))(seg_conv6b)
seg_conv7a = Conv2D(filters=32, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv7a)
seg_conv7a = Add()([seg_conv7a, conv7a])
seg_conv7a = BatchNormalization()(seg_conv7a)
seg_conv7a = Activation('relu')(seg_conv7a)
seg_conv7b = Conv2D(filters=32, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv7a)
seg_conv7b = Add()([seg_conv7b, conv7b])
seg_conv7b = BatchNormalization()(seg_conv7b)
seg_conv7b = Activation('relu')(seg_conv7b)
seg_decoder_output = Conv2D(filters=8, strides=(1, 1), kernel_size=(3, 3), padding='same')(seg_conv7b)
mask_conv4b = Conv2D(filters=512, strides=(1, 1), kernel_size=(3, 3), padding='same')(conv4a)
mask_conv4b = Add()([mask_conv4b, conv4b])
mask_conv4b = BatchNormalization()(mask_conv4b)
mask_conv4b = Activation('relu')(mask_conv4b)
mask_conv4c = Conv2D(filters=512, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv4b)
mask_conv4c = Add()([mask_conv4c, conv4c])
mask_conv4c = BatchNormalization()(mask_conv4c)
mask_conv4c = Activation('relu')(mask_conv4c)
mask_conv4d = Conv2D(filters=256, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv4c)
mask_conv4d = Add()([mask_conv4d, conv4d])
mask_conv4d = BatchNormalization()(mask_conv4d)
mask_conv4d = Activation('relu')(mask_conv4d)
mask_conv5a = UpSampling2D(size=(2, 2))(mask_conv4d)
mask_conv5a = Conv2D(filters=128, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv5a)
mask_conv5a = Add()([mask_conv5a, conv5a])
mask_conv5a = BatchNormalization()(mask_conv5a)
mask_conv5a = Activation('relu')(mask_conv5a)
mask_conv5b = Conv2D(filters=128, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv5a)
mask_conv5b = Add()([mask_conv5b, conv5b])
mask_conv5b = BatchNormalization()(mask_conv5b)
mask_conv5b = Activation('relu')(mask_conv5b)
mask_conv6a = UpSampling2D(size=(2, 2))(mask_conv5b)
mask_conv6a = Conv2D(filters=64, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv6a)
mask_conv6a = Add()([mask_conv6a, conv6a])
mask_conv6a = BatchNormalization()(mask_conv6a)
mask_conv6a = Activation('relu')(mask_conv6a)
mask_conv6b = Conv2D(filters=64, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv6a)
mask_conv6b = Add()([mask_conv6b, conv6b])
mask_conv6b = BatchNormalization()(mask_conv6b)
mask_conv6b = Activation('relu')(mask_conv6b)
mask_conv7a = UpSampling2D(size=(2, 2))(mask_conv6b)
mask_conv7a = Conv2D(filters=32, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv7a)
mask_conv7a = Add()([mask_conv7a, conv7a])
mask_conv7a = BatchNormalization()(mask_conv7a)
mask_conv7a = Activation('relu')(mask_conv7a)
mask_conv7b = Conv2D(filters=32, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv7a)
mask_conv7b = Add()([mask_conv7b, conv7b])
mask_conv7b = BatchNormalization()(mask_conv7b)
mask_conv7b = Activation('relu')(mask_conv7b)
mask_decoder_output = Conv2D(filters=1, strides=(1, 1), kernel_size=(3, 3), padding='same')(mask_conv7b)
model = Model(inputs=input, outputs=[inking_decoder_output, seg_decoder_output, mask_decoder_output])
```

The “input” node is the placeholder for the input image, the “inking_decoder_output” is for the line drawing output, the “seg_decoder_output” is for the regular screentone segmentation output, and the “mask_decoder_output” is for the irregular screen texture mask output.

3.2 Implementation details of previous methods

We present detailed descriptions of the tested previous methods.

3.2.1 Clip Studio Paints [2020]

Clip Studio Paints [2020] supports a vast majority of screentones with commercial standards, *e.g*, dotted screentone, squared screentone, triangle screentone, *etc*. It allows users to choose the scale and type of screen patterns. We configure the software to ensure that it uses the

same type of screen tone as ours and the scales of the patterns are the same. We only use the screentone filter and other settings are remain untouched. No other post-processing filters are applied.

3.2.2 Qu et al. [2008]

Qu et al. [2008] is an optimization-based method that compose screentones and one-pixel-width boundaries to reconstruct images. It can use arbitrary type of screentones and we set it to use the same screentone data with us.

3.2.3 Isola et al. [2017] (Pix2Pix)

Our dataset can be composed to *illustration-to-manga* pairs, and these pairs can be used to train the learning-based Pix2Pix (Isola et al. [2017]) framework. We directly use the official pix2pix implementation to learn such *illustration-to-manga* pairs, and compare it to our frameworks.

3.2.4 Li et al. [2019]

Li et al. [2019] is a learning-based solution to jointly learn line drawing maps and tone maps. Our dataset also contains line drawing maps and tone maps (the tone maps can be composed by blending the pasted regular screentone segmentation and irregular screen texture). We train the framework of Li et al. [2019] using the paired data composed by our dataset, and compare it to our frameworks.

3.2.5 Xie et al. [2020]

Xie et al. [2020] is a learning-based VAE method that directly learn the distribution of manga screentones. We use the same method as we train Pix2Pix to obtain the training data for Xie et al. [2020]. We follow the official implementation details of Xie et al. [2020], and compare it to our frameworks.

4 Copyrights and Ethics Statements

Fig. 15, 17, 21, 29, 35, 36, 37, 40, 55, 56, 59, and 60 are artworks used with artist permission. The artworks may contain commercial elements like the focused Asia-style illustration topics (*e.g*, flower, girl, boy, elf, magic, fantasy, love, prayer, *etc*), Asia-style female/male depiction (*e.g*, romantic, cute, gorgeous, elegant, *etc*), and so on. These artworks captures the real data distribution in the real market for commercial illustrations and manga in recent years. The choice of artworks is fully based on technical merits and unchangeable because those commercial elements are indispensable for our technical presentation to expose the real targets or problems of the commercial manga/illustration that we actually want to assist. The choice of artworks is not on behalf of the researchers' preference or criteria.

References

Software Clip Studio Paints. 2020. Clip Studio Paints. *clip studio* (2020).

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2017).

Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. 2019. Im2Pencil: Controllable Pencil Illustration from Photographs, In IEEE Conference on Computer Vision and Pattern Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*.

Yingge Qu, Wai-Man Pang, Tien-Tsin Wong, and Pheng-Ann Heng. 2008. Richness-Preserving Manga Screening. *ACM Transactions on Graphics (SIGGRAPH Asia 2008 issue)* 27, 5 (December 2008), 155:1–155:8.

Minshan Xie, Chengze Li, Xueting Liu, and Tien-Tsin Wong. 2020. Manga Filling Style Conversion with Screentone Variational Autoencoder. *ACM Transactions on Graphics (SIGGRAPH Asia 2020 issue)* 39, 6 (December 2020), 226:1–226:15.