

# cGAN-based Manga Colorization Using a Single Training Image

Paulina Hensman\*  
 Dept. Information and  
 Communication Eng.  
 University of Tokyo  
 phensman@kth.se

## ABSTRACT

The Japanese comic format known as Manga is popular all over the world. It is traditionally produced in black and white, and colorization is time consuming and costly. Automatic colorization methods generally rely on greyscale values, which are not present in manga. Furthermore, due to copyright protection, colorized manga available for training is scarce.

We propose a manga colorization method based on conditional Generative Adversarial Networks (cGAN). Unlike previous cGAN approaches that use many hundreds or thousands of training images, our method requires only a single colorized reference image for training, avoiding the need of a large dataset.

Colorizing manga using cGANs can produce blurry results with artifacts, and the resolution is limited. We therefore also propose a method of segmentation and color-correction to mitigate these issues. The final results are sharp, clear, and in high resolution, and stay true to the character's original color scheme.

## CCS CONCEPTS

- Computing methodologies → Computer vision tasks; Adversarial learning; Image manipulation; Image segmentation; Neural networks; Interest point and salient region detections;

## KEYWORDS

Manga, colorization, segmentation, GAN, cGAN

## 1 INTRODUCTION

Manga is the main comic style used in Japan, and it is enjoyed by all ages across the world. It is produced at a rapid pace, and is traditionally drawn in black and white. Instead of classic shading, screentones are used to show light and texture.

A colorized manga is often more visually appealing, and with digital distribution methods color printing costs are no longer an issue. That being said, the actual colorization of the image still takes time and requires a skilled artist. Any way to automate this process could potentially be very helpful in production of colorized manga.

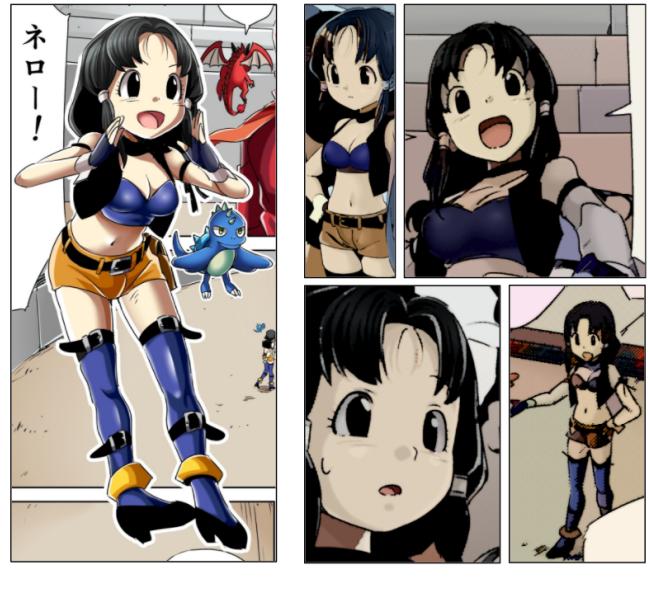
While a fully automatic method for manga colorization would be optimal, some user input will always be required for a correct output. Certain parts of a manga image such as skin color and background elements could possibly be correctly estimated and colorized by a

\*Exchange from Royal Institute of Technology, Sweden

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2017 Copyright held by the owner/author(s).

Kiyoharu Aizawa  
 Dept. Information and  
 Communication Eng.  
 University of Tokyo  
 aizawa@hal.t.u-tokyo.ac.jp



(a) Training image

(b) Our results

**Figure 1:** On the left, a colorized training image. On the right, images colorized by our method. ©Whomor

good method. However, hair color and other attributes that make a character unique can not possibly be inferred from a monochrome image. We therefore propose a method using a colorized image as reference to colorize other images of the same character.

The same characters tend to recur many times in a manga, especially if the manga goes on for several volumes. Additionally, uniforms such as school uniforms, team uniforms, and military uniforms are common. Requiring only a single reference image for each character or each uniform could thus save a lot of time when colorizing a manga. An example of a recurring character is shown in Fig. 1.

Comics are protected by copyright as works of art, making it difficult to gather large datasets of manga to use for training. The largest available dataset is Manga109[10][1], which unfortunately contains only monochrome images. This is a constraint for manga colorization methods, as methods that rely heavily on large amounts of training data can not be used.

We propose a solution for the task of manga colorization using conditional Generative Adversarial Networks (cGANs). Unlike previous cGAN literature, we demonstrate convincing results using a single training image of the character to be colorized. This eliminates the need for a large dataset of colorized manga unprotected

by copyright, while also using the color palette specific to the character during colorization. Our method could potentially be used by manga artists or, with permission, by their fans to colorize images, creating more colorized works for fans of the genre to enjoy.

The contributions of our paper are summarized below.

- We show that a single colorized image is enough to train a cGAN for the task of manga colorization for a specific character.
- We propose a chain of post-processing methods to refine noisily colorized manga images.
- Put together, we introduce a full system for manga colorization. It is the first method for manga colorization that does not require user interaction.
- We use two different datasets, with and without screentones, to show the effectiveness of our method in both cases.

## 2 RELATED WORK

### 2.1 Color transfer methods for natural images

There are many existing methods for color transfer, i.e. using the colors from one image to colorize another[14][6]. The common thread they share is that they make use of preexisting information in the image to be colored, such as color information for a color transfer or luminance in a greyscale image to be colored. Manga is however in its pure form a binary image of only black and white. While shading exists, it comes from screentones which is again just black patterns on white. Thus, the information those methods rely on is not available.

### 2.2 Style transfer methods

In a similar vein to color transfer, style transfer attempts to transfer the style from a reference image onto a new image. Regarding monochrome manga and colorized manga as two different styles, this method could allow us to transform a monochrome manga image into a colorized one. Notable work in this field has been done by Hertzmann et al.[5] and Chuan Li et al.[9].

### 2.3 Conditional Generative Adversarial Networks

The recently introduced Generative Adversarial Networks (GAN)[4] by Goodfellow et al. have shown promising results in generating vividly colorful images[12][13][3]. A notable recent example is the the conditional Generative Adversarial Network (cGAN) presented by Isola et al.[7], which generates new images by conditioning on a target image. This method successfully learned a mapping between edge lines and image for objects such as shoes and handbags. The edge lines are similar to our monochrome manga images, indicating that similar results could be achieved for our task using this method. Up until now, large amounts of training data have been used to train this type of network for generalizing. In the original paper, the smallest training set mentioned consisted of 400 images. We will show that a single colorized reference image is enough to train such a network to colorize a specific character.

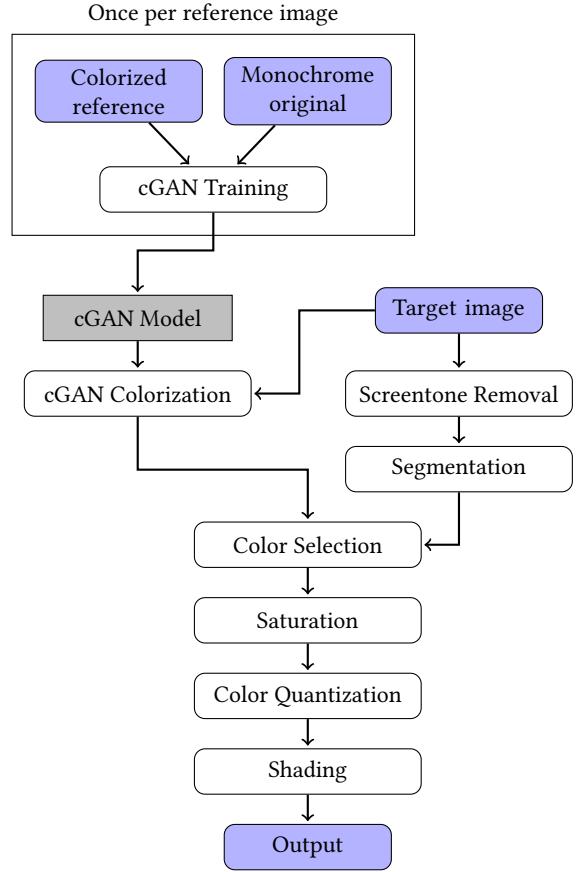


Figure 2: A diagram of the proposed method.

### 2.4 Manga colorization

There have only been a few works on manga colorization. As manga images consist of only black lines and screentones on white background, they are different in nature from natural images, making them more difficult to work with.

Qu et al.[11] proposed a method to colorize manga using hint color scribbles manually added by the user. The method required a lot of user interaction for each image to add the hint colors.

A method for manga colorization based on a reference image was proposed by Sato et al.[15]. Their approach used graph matching to match segments of the reference image to the target image. Their method was limited to simple images.

## 3 METHOD

Our method consists of two main parts, cGAN colorization and post-processing. A diagram of the method is shown in Fig. 2, and a sample output image from each step is shown in Fig. 7. The required training data is a single colorized image and its corresponding monochrome original. Input data is a monochrome target image.

Some parameters have different optimal settings depending on the image, and are thus available for adjustment by the user. An overview of the tunable parameters is shown in Table 1.



Figure 3: A sample page from the Monster Beat dataset, monochrome and colorized.

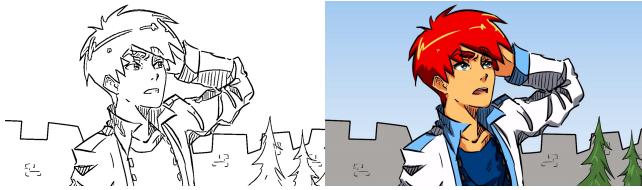


Figure 4: A sample frame from the Morevna dataset [2], with a matching generated line drawing.

### 3.1 Dataset

The data used comes from two sources.

We have used 8 pages of the manga "Monster Beat", where each page has both a colorized and a monochrome version. See Fig. 3 for a sample page. The manga contains 2 prominent characters, one male and one female. The male character appears 18 times in the set, while the female appears 11 times. There is also an older male character appearing 6 times. Appearances include face, half-body, and full-body.

We have also used frames from project Morevna[2], an animated cartoon drawn in manga style. Morevna consists of two videos summing up to a total of 15 minutes, featuring the same two main characters drawn in slightly different styles. At two frames per second, removing all black frames, we created a dataset of 1600 images from the videos. Examples are shown in Fig. 4. As all images were already colorized, we used the screen tone removal method described in section 3.2 to create monochrome line drawings from the images.

### 3.2 Screenshot removal

The screenshot tones used for shading and texture in manga can be helpful or unhelpful when colorizing. In the cGAN colorization step, the screenshot tones work as hints, as areas with the same type of screenshot tend to have the same color. During our segmentation

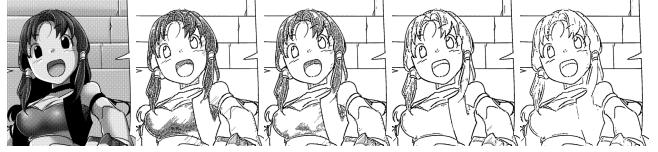


Figure 5: An original monochrome image on the left, followed by the results of tone removal after no Gaussian blur, and after Gaussian blur with a kernel radius of 1, 2 and 3 respectively.

method however, the gaps in the screenshot tones form extra segments, dividing up the true segments. We thus remove the screenshot tones before performing the segmentation.

To extract screenshot tones, we use the method proposed by Ito et al.[8]. However, if a screenshot-free version of the target image is available, which might be the case if the user is the original artist, supplying this version as well gives cleaner lines in the final output.

When working with images in high resolution, screenshot tones will sometimes not be completely removed by Ito et al.'s method. This is presumably due to the sharpness of the screenshot tones causing them to be confused with edge lines. Applying a Gaussian blur to the images before removing the screenshot tones produced better results to a certain extent. As seen in Fig. 5, Gaussian blur with a kernel radius of 1 or 2 gave improved results, while at a kernel radius of 3 the outlines started disappearing, reducing the quality of the result. We can see that there is an optimal value, and our experiments show that it depends on the input image. We therefore make this setting available for adjustment by the user.

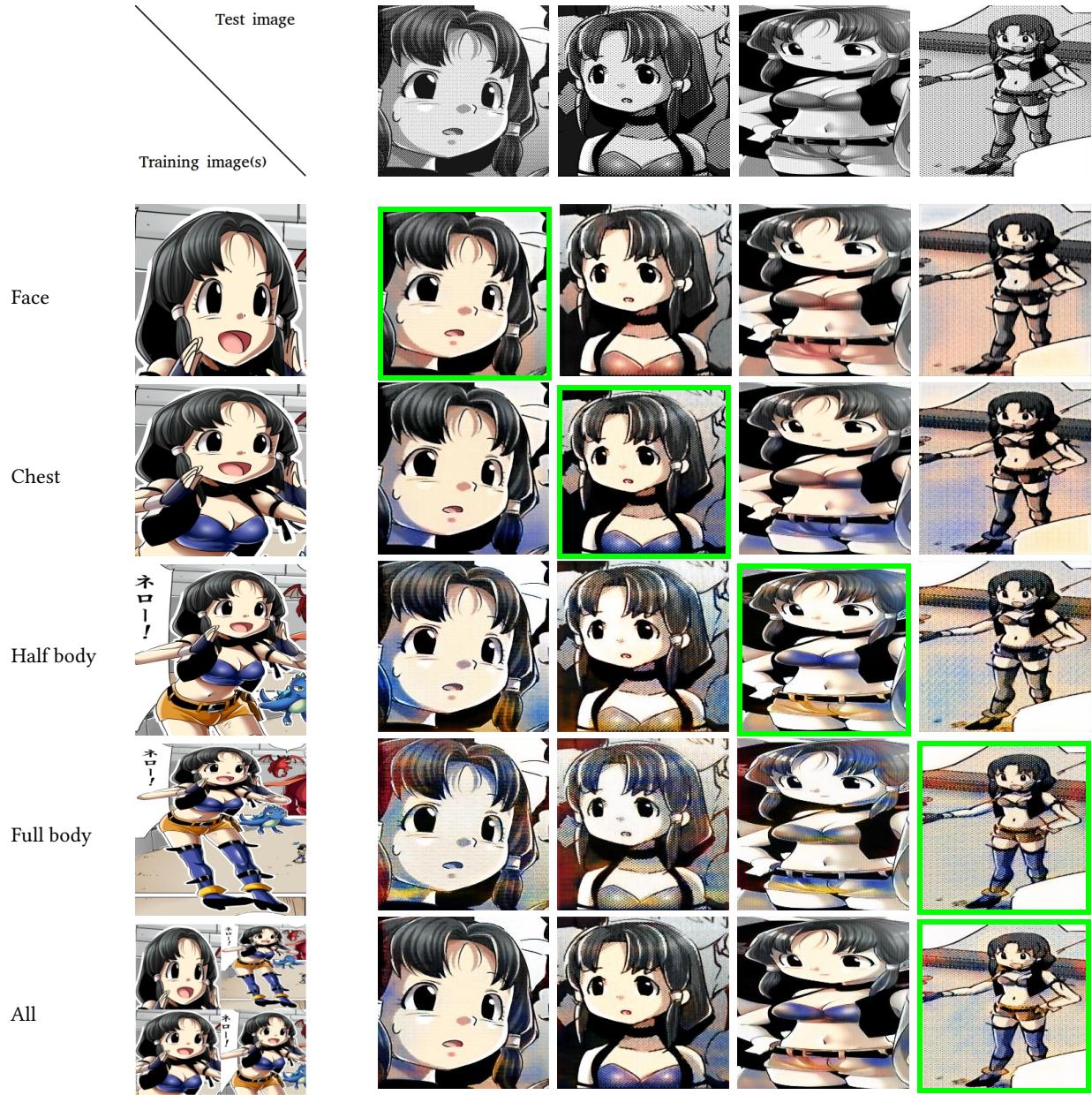
### 3.3 cGAN colorization

We use Isola et al.'s[7] method for an initial colorization of the images. The training is done with a single pair of images: a colorized reference image and a corresponding monochrome image. The monochrome image can be generated from the colorized image if no original is available. If the target images have screenshot tones, converting the colorized image to greyscale is sufficient. Otherwise, Ito et al.'s[8] screenshot removal method can be used to create line drawings as we did for the Morevna dataset.

We found that the coloring stabilizes after training for less than 100 iterations, while further training sharpens the edges of the output image. The results shown in this paper are based on training with 500 iterations, but results after 100-200 iterations are not significantly worse.

As we use only a single training image, this method is in its nature inflexible, which means that it is beneficial to use reference images as similar to the image to be colorized as possible. This does not mean several reference images per character are required, however. Starting with a full body image (or less, if the character never appears in full body) of the character, this image can be cropped to show only face, face and upper body, etc.

Fig. 6 shows results from training with 4 different crops of a reference image on 4 different testing images, plus the results for a network trained with all 4 crops of the image. There is a clear diagonal showing that a crop of only the face is best for colorizing a face, and so on, the exception being the full body image which



**Figure 6: Comparison between combinations of training image and test image for cGAN colorization results.**

we considered a tie between the full body network and the combination network. The combination network was second best in all other categories. Backed by these results, we recommend either training several networks on different crops of the reference image, or training a single network on several crops.

Due to the dataset consisting of only a single image, training is fast, completing 100 iterations in less than 3 minutes on a K80 Tesla GPU. It would not be infeasible to run the training on a CPU.

### 3.4 Segmentation

Manga images for the most part have a natural decomposition into segments, for which each segment has a single color if shading is disregarded. We divide the target image into segments and select a single color for each segment to remove any blur or artifacts created in the cGAN colorization step. For this step, we remove the screentones from the target image, as the spaces formed by the screentones would otherwise form many tiny segments.

We implemented *trapped-ball segmentation* as proposed in [18], with a few changes. The method works by, metaphorically, placing

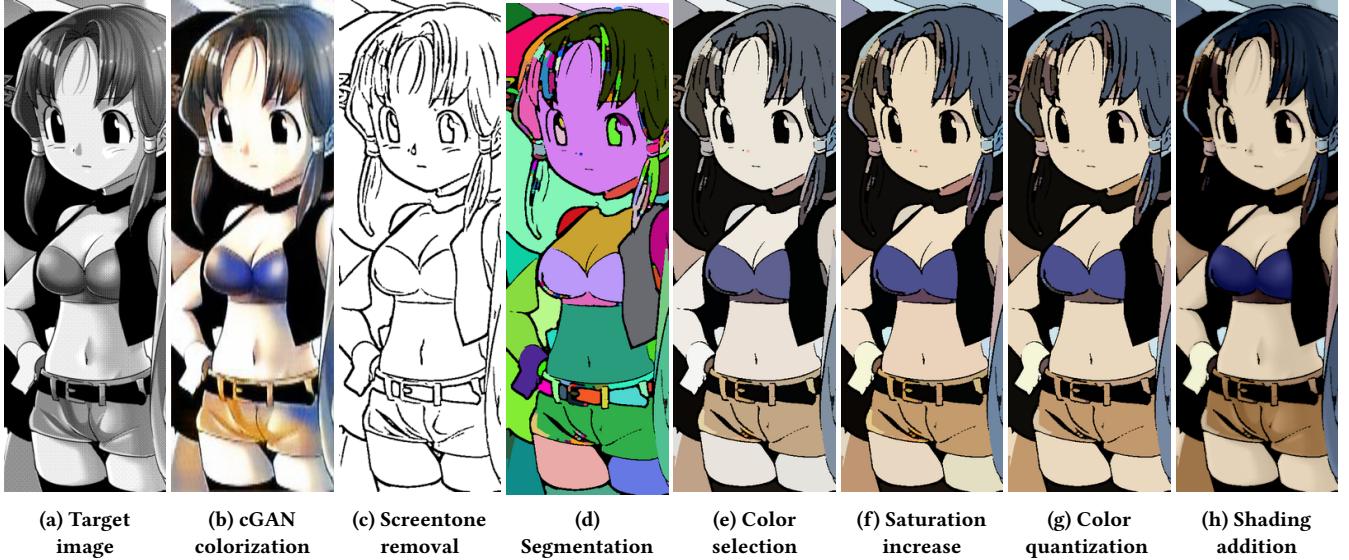


Figure 7: Output from each step in the method.



Figure 8: Segments properly separated by the trapped-ball method despite gaps in the edges.

a ball of a certain diameter in the image. The pixels that the ball can get to without crossing an edge belong to the same segment. This means that any gap between edges narrower than the ball’s diameter will be ignored, preventing leaking. See Fig. 8. This step is repeated by placing the ball in all possible empty areas, then repeated again with smaller balls.

The original method uses a *region growing* step to reach narrow corners, expanding the segments to adjacent pixels with similar color. This step relies on the different segments having different colors, which they do not in our monochrome input images. We therefore skipped this step, and instead expanded all segments to any locations reachable without crossing an edge line after completing the algorithm with ball size 2. Finally, we used a ball of size 1 to segment all remaining areas. Fig. 7d shows a segmented image.

This change in the algorithm makes it more sensitive to the choice of initial ball size, as corners not reached by a larger ball may be filled by a smaller ball before the region growing step is reached. While this encourages smaller starting ball sizes, any gap wider than the starting ball diameter will leak, essentially putting a lower bound on the starting size. If the target image has a few wide gaps, it can be worthwhile to manually close them with a few

quick lines. For some of the results shown in this paper, we have added up to four lines to the target image for this reason.

We leave the choice of initial ball size up to the user, recommending a value between 2-5 pixels.

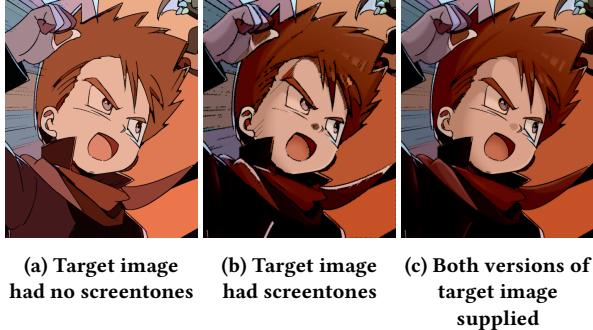
### 3.5 Colorization

The remaining colorization and post-processing happens in 4 steps.

**3.5.1 Select color per segment.** We enlarge the cGAN output to the same dimensions as the target image. For each segment found in the previous step, the mean color of the corresponding pixels of the cGAN image in RGB space is selected as the color for the entire segment. Two other selection methods were considered: selecting the most prominent color in the segment, and selecting one of the most common colors from the cGAN output image. In the end however, selecting the mean color gave good results at a low level of complexity.

**3.5.2 Increase saturation.** The averaging in the previous step can sometimes render the colors slightly washed out compared to the training image, due to colors in the cGAN colorization output floating out between segments. To combat this, saturation can be increased in the HSV colorspace. The optimal amount varies by image, and can be left up to the user. We found that values in the range 5-10% produced satisfactory results. This step also helps separate the colors in RGB space for the next step.

**3.5.3 Color quantization.** To remove any remaining noise, color quantization can be performed to reduce the number of unique colors in the image. In Fig. 7g, for example, the color quantization step consolidates the different shades of the character’s legs to the same color.  $k$  clusters of colors are found using k-means clustering in RGB space, and all pixels are colorized with the center value of their cluster. The optimal number of colors  $k$  can differ between different images, so we leave it up to the user. The optimal number



**Figure 9: Results using different versions of the target image.**

Parameter	Permissible range	Recommended range
Gaussian blur radius	$> 0$	1 - 2
Starting ball size	$> 1$	2 - 5
Saturation increase	$< 255$	10 - 25
Color clusters	$> 0$	5-20

**Table 1: Tunable parameters and their ranges.**

tends to lie in the interval 5-12. In many cases, this step is not necessary.

**3.5.4 Shading.** Finally, if the input image has screentones, we can use these to apply shading to the image. To achieve this, we first apply a strong Gaussian blur to the input image, smoothing out the sharp screentones. We saw good results using a kernel radius two pixels bigger than the one used for screentone removal in Section 3.2. We then darken the appropriate areas of the colorized image using the following on each pixel in each of the RGB channels:

$$C_{[i,j]} = C_{[i,j]} - \frac{255 - M_{[i,j]}}{3},$$

where  $C$  is the colorized image we are working on and  $M$  is the blurred monochrome input image.

## 4 RESULTS

Our results are shown in Fig. 1, 9, 10, and 11.

In the cGAN colorization step, high-level structures such as hair, skin, and clothes are recognized and assigned the correct color in a translation-invariant manner. Remaining inaccuracies are mitigated by our subsequent post-processing methods. Comparing the cGAN output on the bottom left in Fig. 13 to the output after post-processing in Fig. 11b demonstrates the effectiveness of our segmentation technique. While the cGAN output's colors float out outside the edge lines, our post-processing confines the colors to their appropriate segments and produces a clean output.

As we showed in Fig. 6, training the cGAN on a cropped version of the training image matching the target image improves the results. If all colors expected in the target image are not present in the training image, the cGAN will not have enough information to colorize it correctly (upper triangle). On the other hand, if more colors are present in the training image than are expected in the

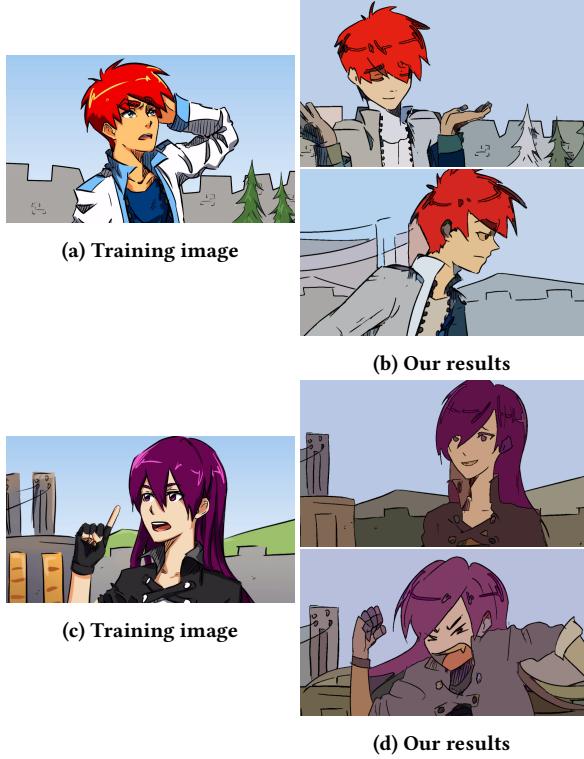


**Figure 10: Our results for Monster Beat dataset. On the left a colorized training image. On the right, images colorized by our method.**

target image, the cGAN tries to fit those colors in the image anyway, leading to unwanted colors in the result (lower triangle).

When the target image has the same type of screentones as the training image, the colorization gets much more accurate, as it allows a mapping between screentone and color. In this way, the screentones help the colorization in much the same way as greyscale coloring would. When the target image does not have screentones, the quality of the results decrease. Example results from the Morevna dataset in Fig. 11 show that certain structures, such as hair, sky, and to some extent hands, were easily identified, but other parts were more problematic. For the first training image, we noted overfitting for a dark blue color in the lower middle of the image, but saw that the red hair was accurately colorized regardless of position in the image.

In many of our results, the edge lines are slightly rough and extra lines have appeared around shaded areas. This is a side effect of the screentone removal step. In cases where the target image lacked



**Figure 11:** Our results for the Morevna dataset.

screentones to begin with, such as in Fig. 11, this phenomenon is not present. However, in those cases there can be no shading. If both versions of the target image is supplied, i.e. with and without screentones, the screentones can be used for shading while keeping the clean lines from the screentone-free version. A comparison is shown in Fig. 9.

#### 4.1 Segmentation

The *trapped ball* segmentation approach worked reasonably well. The resistance to small gaps in the edge lines proved useful when working with our dataset. It did however give rise to issues in narrow areas with gaps. After all areas that could fit a *ball* with a diameter of 2 pixels were assigned, all existing segments got expanded to all non-assigned pixels reachable without crossing an edge line, referred to as the *region expansion* step. This made larger segments float into areas that were too narrow to have been filled, but that had gaps in their outline. This was common in for example pointy strands of hair, as seen in Fig. 12. Despite this, we opted to use the region growing step, as forgoing it gave rise to tiny segments being formed in all corners of the ground truth segments, resulting in a more mottled colorization.

The segmentation also had troubles in areas where the screentone removal was not fully effective, forming small dots of different colors in an area that should have been uniformly colored. An example is again in Fig. 12.



**Figure 12:** Fail case for segmentation: leaking in the tip of the hair on top right, and extra segments due to remaining screentones in lower middle.



**Figure 13:** Comparison between cGAN colorization (left) and style transfer (right) output.

#### 4.2 Comparison with Style transfer

We compare our cGAN colorization results to that of neural network-based style transfer. We use a combination of [5] and [9] as implemented by Wentz[17], using the VGG16 features[16] for matching instead of looking at pixels directly. We keep the screentones in the images for this method. Results from this method are compared to our results in Fig. 13.

While the style transfer method is also mostly correct in color placement, especially with screentones present for guidance, the colors produced are pale. When no screentones are present, as in the image from the Morevna dataset on the bottom, our method was much more successful in locating important areas such as hair and skin.

The greatest downside of the style transfer method is that each pair of images requires a long time to process, normally at least 5 minutes. To colorize  $n$  images of a character's face, the style transfer method would need to be run  $n$  times while our method would only need to be trained once. If the task is colorizing an entire manga volume, where a single character can easily appear over a hundred times, our method is much faster.

## 5 CONCLUSION

We have shown a method for colorizing manga images, requiring only a single colorized reference image for training. The results are clear, sharp and in high resolution, and stay true to the character's original color scheme.

Our method trains a cGAN on a single colorized reference image of a manga character, and uses the model to colorize new images of the same character. Blur, ambiguities and other problems in the output are mitigated by a segmentation and color-correction method. In many cases, the results of the cGAN colorization are good even as stand-alone results.

While the current training time is very fast, it might be possible to get even faster results by using a cGAN pre-trained on a slightly larger dataset and finetuning it on the character-specific reference images. For a user wanting to colorize many different characters, this could save some extra time. This would be interesting to explore in future work.

For manga colorization, the biggest constraint is often the lack of available data, due to copyright restrictions. Methods using very low amounts of training data are thus very important in this field. By using training data featuring the same character as the target images, our method produced compelling results using only a single training image.

## ACKNOWLEDGMENTS

The authors would like to thank Whomor for the use of their images.

## REFERENCES

- [1] Manga 109, 2015 <http://www.manga109.org>
- [2] Morevna Project, 2016, <https://morevnaproject.org/>
- [3] Alexey Dosovitskiy, Thomas Brox, Generating images with perceptual similarity metrics based on deep networks, Advances in Neural Information Processing Systems, pp.658-666, 2016.
- [4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, Generative adversarial nets, Advances in Neural Information Processing Systems, pp.2672-2680, 2014.
- [5] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin, Image analogies, Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM, pp.327-340, 2001
- [6] Hristina Hristova, Olivier Le Meur, Remi Cozot, and Kadi Bouatouch, Style-aware robust color transfer, EXPRESSIVE International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging, 2015.
- [7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros, Image-to-image translation with conditional adversarial networks, arXiv preprint arXiv:1611.07004, 2016.
- [8] Kota Ito, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa, Separation of Manga Line Drawings and Screentones, Eurographics, pp.78-76, 2015
- [9] Chuan Li and Michael Wand Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis, CoRR, abs/1601.04589, 2016.
- [10] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa, Sketch-based manga retrieval using manga109 dataset, Multimedia Tools and Applications, Springer, pp.1-28, 2016.
- [11] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng Manga colorization, ACM Transactions on Graphics (TOG), vol.25, no.3, pp.1214-1220, 2006.
- [12] Alec Radford, Luke Metz, and Soumith Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434, 2015.
- [13] Scott Reed, Zeynep Akata, Xincheng Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee, Generative adversarial text to image synthesis, Proceedings of The 33rd International Conference on Machine Learning, vol.3, 2016.
- [14] Erik Reinhard, Michael Adhikmin, Bruce Gooch, and Peter Shirley, Color transfer between images, IEEE Computer graphics and applications, vol.21, no.5, pp.34-41, 2001.
- [15] Kazuhiro Sato, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa, Reference-based manga colorization by graph correspondence using quadratic programming, ACM SIGGRAPH Asia 2014 Technical Briefs, 2014.
- [16] K. Simonyan and A. Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR, abs/1409.1556, 2014.
- [17] Adam Wentz, Neural Image Analogies, GitHub, <https://github.com/awentzonline/image-analogies>.
- [18] Song-Hai Zhang, Tao Chen, Yi-Fei Zhang, Shi-Min Hu, and Ralph R Martin, Vectorizing cartoon animations, IEEE Transactions on Visualization and Computer Graphics, vol.15, no.4, pp.618-629, 2009.