Correspondence Neural Network for Line Art Colorization

Trung D.Q. Dang
Cinnamon AI
tyler@cinnamon.is

Thien Do
Cinnamon AI
hades@cinnamon.is

Anh Nguyen
Cinnamon AI
cat@cinnamon.is

Van Pham
Cinnamon AI
kan@cinnamon.is

Quoc Nguyen
Cinnamon AI
akari@cinnamon.is

Bach Hoang
Cinnamon AI
gale@cinnamon.is

Giao Nguyen
Cinnamon AI
enzo.nguyen@cinnamon.is

KEYWORDS

colorization, neural network, component matching

ACM Reference Format:
Trung D.Q. Dang, Thien Do, Anh Nguyen, Van Pham, Quoc Nguyen, Bach Hoang, and Giao Nguyen. 2020. Correspondence Neural Network for Line Art Colorization. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters (SIGGRAPH '20 Posters), August 17, 2020.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3388770.3407418

1 INTRODUCTION

Line art colorization is a critical process in animation production, but this procedure is usually tedious and time-consuming. There are hint-based colorization tools [LvMin Zhang and Liu 2018], but they fail to transfer colors to a new sketch without user intervention. GAN-based style transfer methods [Goodfellow et al. 2014] focus on style rather than correct colors, and suffer from color bleeding. GAN models are also data hungry, since they need lots of training examples to generalize, which are not readily available in animation industry. In addition, color propagation methods [Meyer et al. 2018] target mainly grayscale photos. Compared to this kind of data, line arts lack texture information.

In this work, we propose a deep architecture to colorize line arts based on a reference image by finding corresponding (connected) components. The proposed method is inspired by Universal Correspondence Network [Choy et al. 2016]. Here, we apply U-net [Ronneberger et al. 2015] for feature extraction on component level instead of pixel level. After decomposing a line art into a set of components, we extract, for each component, an embedding vector by a network. These embeddings are used to determine component correspondences between the reference and a target line art, so reference colors can be propagated onto corresponding regions.

2 OUR APPROACH

The input to our solution consists of a color reference image I, its line art version A and a target line art B. Practically, drawing

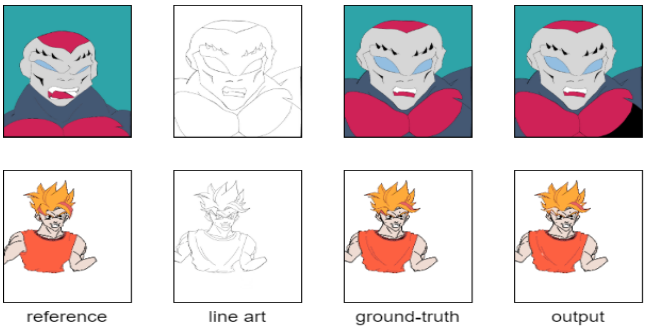

Figure 1: The sample output of our colorization method.

closed continuous sketch strokes is not natural to artists, but non-continuous strokes lead to poor performance of component extraction. Thus, our first step relies on automatic correction of components with open contours in the line arts. We use Fourey's algorithm [Fourey et al. 2018] to detect all endpoints of non-continuous strokes in the image and connect them based on their orientation.

Then, we extract connected components of line arts A and B, as well as component label maps L_A and L_B. We compute Hu moments of each component as shape features, and propagate these moments onto component regions in L_A and L_B to get input feature maps F_A and F_B. The features at a point are set to be the shape features of its parent component.

$$F_A(x, y) = Hu(A_i) \quad \text{if} \quad L_A(x, y) = i \tag{1}$$

where (x, y) is a point in component i and A_i is the shape of component i. The purpose is to provide more context for further steps, instead of black-and-white input.

In the next phase, the shape feature map of each line art (F_A or F_B) is fed to a U-net (Figure 2). We use CoordConv [Liu et al. 2018] for location information. We need to learn high-level context, such as relations between components, but we don't want to stack many layers to the U-net and increase the processing time. Instead, we keep the U-shape structure shallow, and add more convolution blocks to the bottom of the U-net. The resulting feature map will be upsampled and concatenated to the last layer of U-net. We compute the final feature map by another convolution block to merge high-level and low-level features from two branches. After computing this feature map, we do average pooling over these features according to the component label map (L_A or L_B) to get embeddings for

SIGGRAPH '20 Posters, August 17, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7973-1/20/08.
https://doi.org/10.1145/3388770.3407418

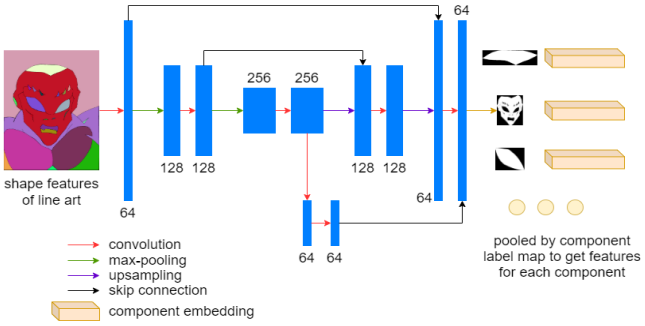

Figure 2: Network architecture.

each component.

$$S_A(i) = \frac{1}{N_i} \sum_{L_A(x,y)=i} G(F_A)(x,y) \tag{2}$$

where (x, y) is a point in component i, G is our network and N_i is the size of component i. This operation is implemented by matrix multiplication between the output feature map and the component label map. The output of this step is two sets of embedding vectors S_A (from F_A and L_A) and S_B (from F_B and L_B).

The output embeddings S_A and S_B are used for component correspondences. We compute pair-wise cosine distance between component embeddings in S_A and S_B. For each component in S_B, we match with a component in S_A where distance is minimum. After finding all matches, we propagate the color of each reference component in I to the corresponding region in the target line art and get the final colorized output.

We train the network by triplet loss [Schroff et al. 2015]. The goal of this loss function is to minimize the distances between matching components and penalize low distances between different components. Given an anchor u in the target line art, its corresponding reference component v_p and a negative component v_n, the loss is defined by:

$$Loss(u) = d(u, v_p) + max(0, K - d(u, v_n)) \tag{3}$$

where d is cosine distance and K is margin. We use color information and hand-crafted features in two ground truth pictures to generate positive component matches. As we know the color of each component, we can keep only certainly correct matches. On the other hand, negative matches are chosen by hard example mining.

3 RESULTS

Our training dataset has 134 shots with 1146 images, and our test dataset has 90 shots with 640 images. We show the results of our evaluation in Table 1.

Table 1: Evaluation results.

	acc-component	processing time
hand-crafted features	63.66% ± 19.96	9.51s ± 20.29
ours	81.16% ± 23.27	11.05s ± 4.64
ours (traditional U-net)	79.61% ± 25.32	12.43s ± 4.71

We display an example of our output in Figure 1. Our first key finding is that the network can learn useful embeddings for component correspondences. As we see, most components are colorized correctly, even when their motions across frames are noticeable.

We compare our network architecture with a traditional U-net with more layers, and more filters at each layer. Although this baseline network has more parameters, its accuracy is not as good as our proposed architecture. We conclude, for our second key finding, our architecture is better for this problem, where global context is more important than fine details.

4 CONCLUSION

In this work, we propose a line art colorization method that combines hand-crafted feature extraction and neural networks to match components between two line arts, so that reference colors can be propagated onto target regions. We conduct experiments and demonstrate the effectiveness of our method, compared to using only computer vision techniques.

For future work, we want our method to be less dependent on continuous sketch strokes, by matching on pixel level instead of component level.

ACKNOWLEDGMENTS

This research was supported by Cinnamon AI. We thank members from Cinnamon AI who provided insight and expertise that greatly assisted the research.

REFERENCES

Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. 2016. Universal correspondence network. In *Advances in Neural Information Processing Systems*. 2414–2422.

Sébastien Fourey, David Tschumperlé, and David Revoy. 2018. A fast and efficient semi-guided algorithm for flat coloring line-arts.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. 2018. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*. 9605–9616.

Tien-Tsin Wong Yi Ji LvMin Zhang, Chengze Li and ChunPing Liu. 2018. Two-stage Sketch Colorization. *ACM Transactions on Graphics* 37, 6 (Nov. 2018). https://doi.org/10.1145/3272127.3275090

Simone Meyer, Victor Cornillère, Abdelaziz Djelouah, Christopher Schroers, and Markus Gross. 2018. Deep video color propagation. *arXiv preprint arXiv:1808.03232* (2018).

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 815–823.