Seamless Manga Inpainting with Semantics Awareness: Supplementary Material

MINSHAN XIE, The Chinese University of Hong Kong
MENGHAN XIA, The Chinese University of Hong Kong
XUETING LIU, Caritas Institute of Higher Education
CHENGZE LI, Caritas Institute of Higher Education
TIEN-TSIN WONG, The Chinese University of Hong Kong

ACM Reference Format:
Minshan Xie, Menghan Xia, Xueting Liu, Chengze Li, and Tien-Tsin Wong. 2021. Seamless Manga Inpainting with Semantics Awareness: Supplementary Material. *ACM Trans. Graph.* 40, 4, Article 96 (August 2021), 7 pages. https://doi.org/10.1145/3450626.3459822

1 ADDITIONAL RESULTS

For the following results in Fig. 1 and Fig. 2: (a) Input, (b) Photoshop[2021], (c) [Barnes et al. 2009], (d) [Yu et al. 2018], (e) [Zeng et al. 2019], (f) [Nazeri et al. 2019], (g) [Xie et al. 2020], and (h) Ours. The mask is labeled in green.

To present our qualitative results in a comprehensive manner, typical inpainted examples with various quality levels are demonstrated in Fig. 3. Specifically, they are ordered with increasing LPIPS scores, indicating decreasing visual quality. The masks are labeled in green. The LPIPS score is shown under each pair of images (masked target image and inpainted image).

We also further illustrate more results for the ablation studies in Fig. 4, Fig. 5 and Fig. 6.

2 DETAILED NETWORK ARCHITECTURE

We adopt the 5-layer discriminator design of PatchGAN in both semantic inpainting network and appearance synthesis network. We introduce a Gaussian noise map as an extra input for both semantic inpainting network and appearance synthesis network. Below shows the network details we adopted in our paper.

2.1 Semantic Inpainting Network

The semantic inpainting network consists of a feature extractor and two branches to separately generate the structural lines and ScreenVAE map. Table 1 listed the detail network architecture of semantic inpainting network G_{inp} and Table 2 shows the architecture of the discriminator D_{ls}.

2.2 Appearance Synthesis Network

The appearance synthesis network follows the encoder-decoder architecture with a contextual attention module[Yu et al. 2018] to borrowing the features with highest similarity to the disoccluded regions. The encoder part has two branch, the image branch and attention branch. Table 3 listed the detail network architecture of G_{syn}. Table 4 shows the network architecture of the discriminator D_{mg}.

REFERENCES

2021. Photoshop. https://www.photoshop.com.
Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, Vol. 28. ACM, 24.
Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. 2019. Edgeconnect: Generative image inpainting with adversarial edge learning. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*.
Minshan Xie, Chengze Li, Xueting Liu, and Tien-Tsin Wong. 2020. Manga filling style conversion with screentone variational autoencoder. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.
Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2018. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5505–5514.
Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. 2019. Learning pyramid-context encoder network for high-quality image inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1486–1494.

Authors' addresses: Minshan Xie, The Chinese University of Hong Kong, msxie@cse.cuhk.edu.hk; Menghan Xia, The Chinese University of Hong Kong, mhxia@cse.cuhk.edu.hk; Xueting Liu, Caritas Institute of Higher Education, tliu@cihe.edu.hk; Chengze Li, Caritas Institute of Higher Education, czli@cihe.edu.hk; Tien-Tsin Wong, The Chinese University of Hong Kong, ttwong@cse.cuhk.edu.hk.

0730-0301/2021/8-ART96 $15.00
https://doi.org/10.1145/3450626.3459822

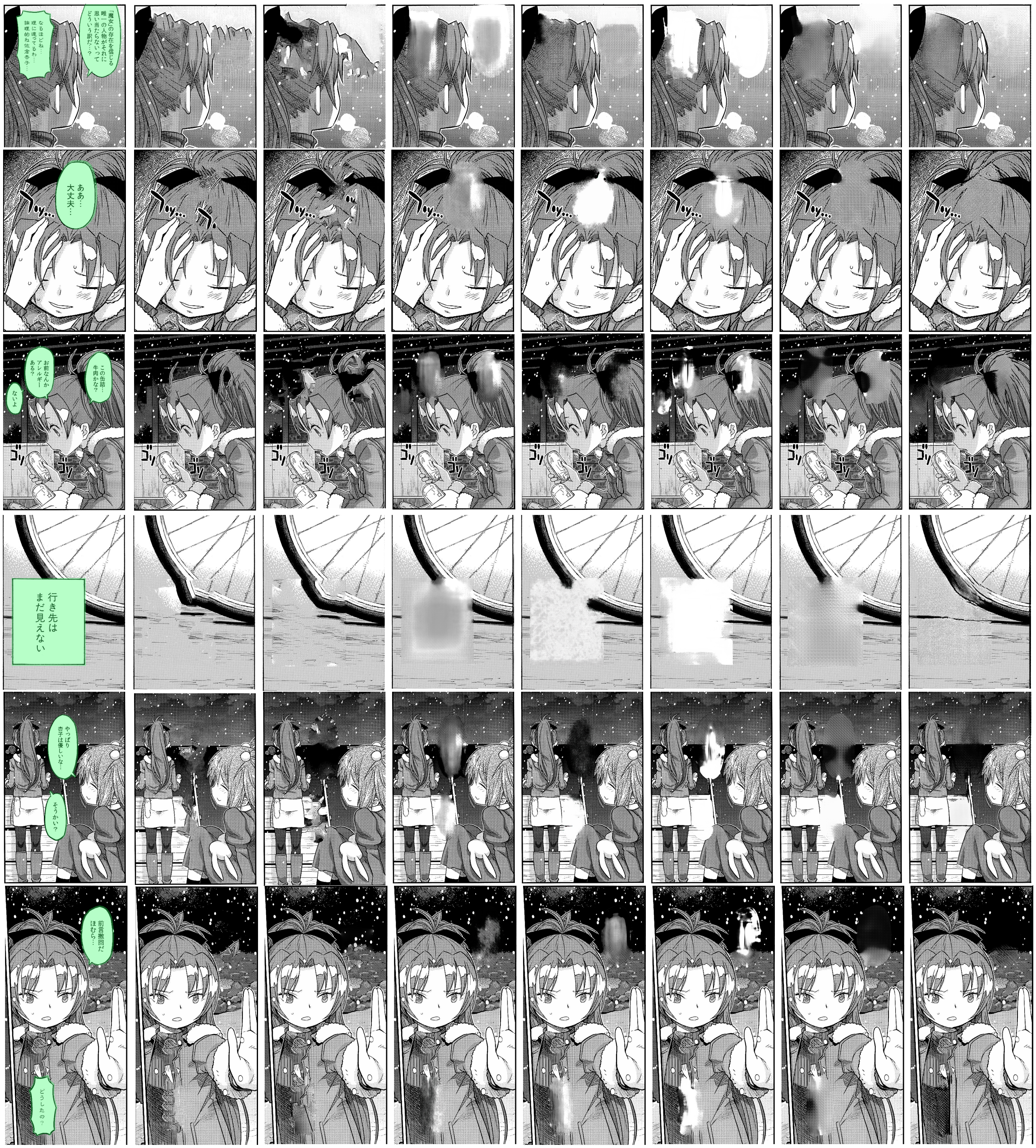


Fig. 1. More inpainted results generated by various state-of-the-arts inpainting methods. KaerimichiNoMajo ©A-10, "Give My Regards to Black Jack" ©Shuho Sato

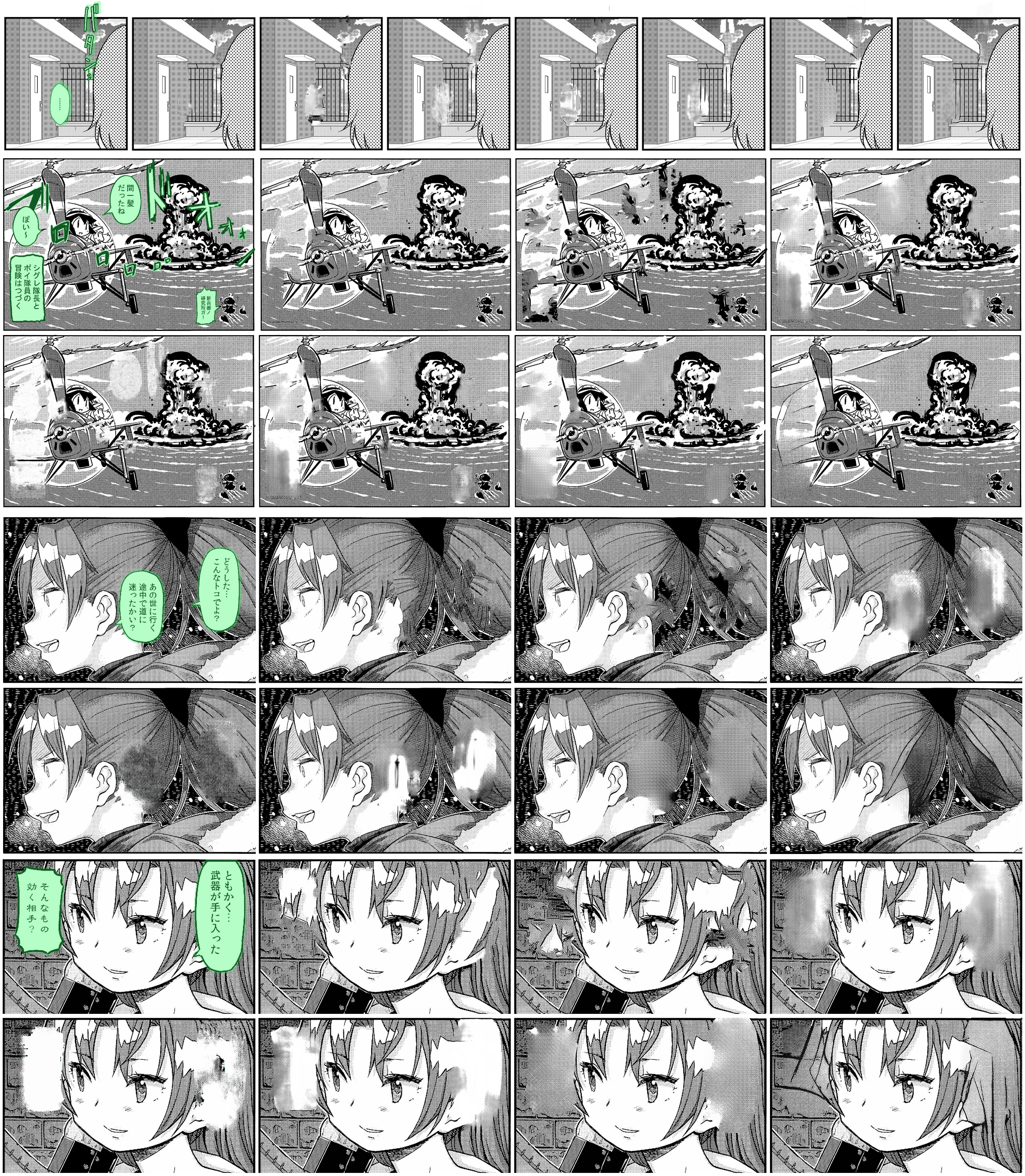


Fig. 2. More inpainted results generated by various state-of-the-arts inpainting methods. "She Great" ©SEIRAN, KaerimichiNoMajo ©A-10

Table 1. Detail network architecture of semantic inpainting network G_{inp}.

Part	Input → Output Shape	Layer Operations
Feature Extractor	$(h, w, 6) \rightarrow (h, w, 64)$	CONV-(N64,K7x7,S1,P3),LN,ReLU
	$(h, w, 64) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	CONV-(N128,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	CONV-(N256,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	CONV-(N256,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	Residual Block: CONV-(N256,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	Residual Block: CONV-(N256,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	Residual Block: CONV-(N256,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	Residual Block: CONV-(N256,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	Residual Block: CONV-(N256,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	Residual Block: CONV-(N256,K3x3,S1,P1),LN,ReLU
Branch_**L**	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	DECONV-(N256,K4x4,S2,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	DECONV-(N128,K4x4,S2,P1),LN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (h, w, 64)$	DECONV-(N64,K4x4,S2,P1),LN,ReLU
	$(h, w, 64) \rightarrow (h, w, 1)$	CONV-(N1,K7x7,S1,P3),LN,ReLU
Branch_**S**	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{4}, \frac{w}{4}, 256)$	UPSAMPLE(2),CONV-(N256,K3x3,S1,P1),ReLU
	$(\frac{h}{4}, \frac{w}{4}, 256) \rightarrow (\frac{h}{2}, \frac{w}{2}, 128)$	UPSAMPLE(2),CONV-(N128,K3x3,S1,P1),ReLU
	$(\frac{h}{2}, \frac{w}{2}, 128) \rightarrow (h, w, 64)$	UPSAMPLE(2),CONV-(N64,K3x3,S1,P1),ReLU
	$(h, w, 64) \rightarrow (h, w, 4)$	CONV-(N1,K7x7,S1,P3),ReLU

Table 2. Detail network architecture of the discriminator D_{ls}.

Part	Input → Output Shape	Layer Operations
Down-sampling	$(h, w, 6) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64,K3x3,S2,P1),LeakyReLU(0.2)
	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128,K3x3,S2,P1),LeakyReLU(0.2)
	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	CONV-(N256,K3x3,S2,P1),LeakyReLU(0.2)
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{16}, \frac{w}{16}, 512)$	CONV-(N512,K3x3,S2,P1),LeakyReLU(0.2)
	$(\frac{h}{16}, \frac{w}{16}, 512) \rightarrow (\frac{h}{16}, \frac{w}{16}, 1)$	CONV-(N1,K3x3,S1,P1),LeakyReLU(0.2)
Output Layer	$(\frac{h}{16}, \frac{w}{16}, 1) \rightarrow (\frac{h}{16}, \frac{w}{16}, 1)$	Sigmoid

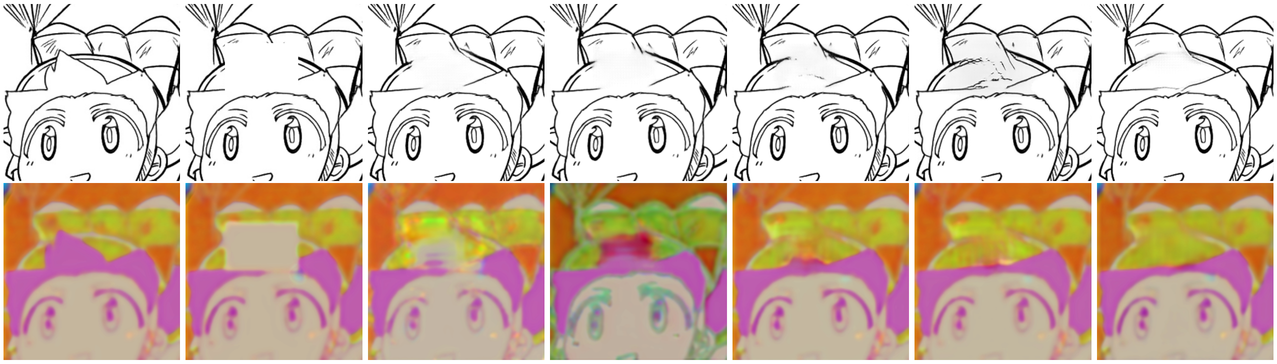
Table 3. Detail network architecture of appearance synthesis network G_{syn}.

Part	Input → Output Shape	Layer Operations
Branch_**I**	$(h, w, 3) \rightarrow (h, w, 32)$	CONV-(N32,K5x5,S1,P2),LN,ReLU
	$(h, w, 32) \rightarrow (\frac{h}{2}, \frac{w}{2}, 32)$	CONV-(N32,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 32) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 64)$	CONV-(N64,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONV-(N128,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONV-(N128,K3x3,S1,P1,D2),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONV-(N128,K3x3,S1,P1,D4),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONV-(N128,K3x3,S1,P1,D8),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONV-(N128,K3x3,S1,P1,D16),LN,ReLU
Branch_**A**	$(h, w, 5) \rightarrow (h, w, 32)$	CONV-(N32,K5x5,S1,P2),LN,ReLU
	$(h, w, 32) \rightarrow (\frac{h}{2}, \frac{w}{2}, 32)$	CONV-(N32,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 32) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 64)$	CONV-(N64,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONV-(N128,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	ContextualAttention(Feature_**I**, Feature_**A**, Mask)
	$(\frac{h}{8}, \frac{w}{8}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONV-(N128,K3x3,S1,P1,D2),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONV-(N128,K3x3,S1,P1,D4),LN,ReLU
Decoder	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{8}, \frac{w}{8}, 128)$	CONCAT,CONV-(N128,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{8}, \frac{w}{8}, 128) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	DECONV-(N128,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{4}, \frac{w}{4}, 64)$	CONV-(N64,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{4}, \frac{w}{4}, 64) \rightarrow (\frac{h}{2}, \frac{w}{2}, 32)$	DECONV-(N32,K3x3,S2,P1),LN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 32) \rightarrow (\frac{h}{2}, \frac{w}{2}, 32)$	CONV-(N32,K3x3,S1,P1),LN,ReLU
	$(\frac{h}{2}, \frac{w}{2}, 32) \rightarrow (h, w, 1)$	CONV-(N1,K5x5,S1,P2),LN,ReLU

Table 4. Detail network architecture of the discriminator D_{mg}.

Part	Input → Output Shape	Layer Operations
Down-sampling	$(h, w, 6) \rightarrow (\frac{h}{2}, \frac{w}{2}, 64)$	CONV-(N64,K3x3,S2,P1),LeakyReLU(0.2)
	$(\frac{h}{2}, \frac{w}{2}, 64) \rightarrow (\frac{h}{4}, \frac{w}{4}, 128)$	CONV-(N128,K3x3,S2,P1),LeakyReLU(0.2)
	$(\frac{h}{4}, \frac{w}{4}, 128) \rightarrow (\frac{h}{8}, \frac{w}{8}, 256)$	CONV-(N256,K3x3,S2,P1),LeakyReLU(0.2)
	$(\frac{h}{8}, \frac{w}{8}, 256) \rightarrow (\frac{h}{16}, \frac{w}{16}, 512)$	CONV-(N512,K3x3,S2,P1),LeakyReLU(0.2)
	$(\frac{h}{16}, \frac{w}{16}, 512) \rightarrow (\frac{h}{16}, \frac{w}{16}, 1)$	CONV-(N1,K3x3,S1,P1),LeakyReLU(0.2)
Output Layer	$(\frac{h}{16}, \frac{w}{16}, 1) \rightarrow (\frac{h}{16}, \frac{w}{16}, 1)$	Sigmoid

Fig. 3. More inpainted results generated by our manga inpainting method. KaerimichiNoMajo ©A-10, "She Great" ©SEIRAN, "Give My Regards to Black Jack" ©Shuho Sato



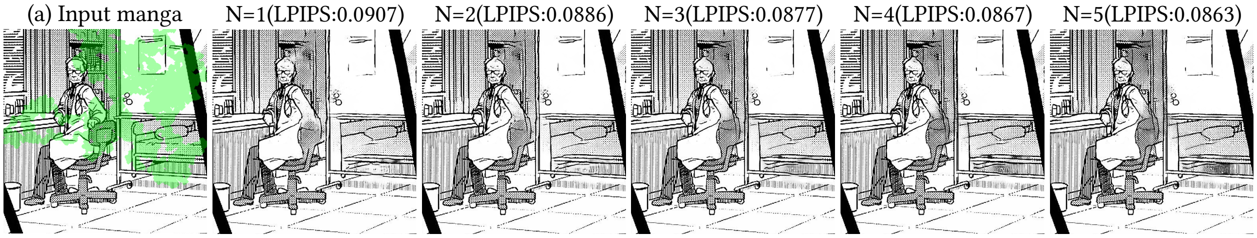
(a) Input (b) Masked input (c) w/o $\mathcal{L}_{\text{lrec}}$ (d) w/o $\mathcal{L}_{\text{srec}}$ (e) w/o $\mathcal{L}_{\text{adv}}$ and $\mathcal{L}_{\text{fm}}$ (f) w/o $\mathcal{L}_{\text{bin}}$ (g) Ours

Fig. 4. The inpainted results generated by semantic inpainting model with and without different loss terms. The ScreenVAE map is visualized after PCA. "She Great" ©SEIRAN

(a) Manga (b) Structural line (c) ScreenVAE map (d) w/o $\mathcal{L}_{\text{mrec}}$ (e) w/o $\mathcal{L}_{\text{srec}}$ (f) w/o $\mathcal{L}_{\text{adv}}$ (g) w/o $\mathcal{L}_{\text{bin}}$ (h) Ours

Fig. 5. The inpainted results generated by the appearance synthesis model with and without different loss terms. The mask is labeled in green. The ScreenVAE map is visualized after PCA. KaerimichiNoMajo ©A-10

(a) Input manga N=1(LPIPS:0.0907) N=2(LPIPS:0.0886) N=3(LPIPS:0.0877) N=4(LPIPS:0.0867) N=5(LPIPS:0.0863)



(b) Input manga N=1(LPIPS:0.1267) N=2(LPIPS:0.1243) N=3(LPIPS:0.1244) N=4(LPIPS:0.1247) N=5(LPIPS:0.1286)

Fig. 6. More inpainted results generated by semantic inpainting model with different iterations. "Give My Regards to Black Jack" ©Shuho Sato