Iterative Predictor-Critic Code Decoding for Real-World Image Dehazing

Jiayi Fu[1] Siyu Liu[1] Zikun Liu[3] Chun-Le Guo[1,2]
Hyunhee Park[4] Ruiqi Wu[1] Guoqing Wang[5] Chongyi Li[1,2*]
[1]VCIP, CS, Nankai University [2]NKIARI, Shenzhen Futian
[3]Samsung R&D Institute China-Beijing [4]CIG, Samsung Electronics
[5]Donghai Laboratory, Zhoushan, Zhejiang
{fujiayi,liusiyu29}@mail.nankai.edu.cn, zikun.liu@samsung.com,
guochunle@nankai.edu.cn, inextg.park@samsung.com,
wuruiqi@mail.nankai.edu.cn, gqwang0420@hotmail.com, lichongyi@nankai.edu.cn

Abstract

This supplementary material presents the network architectures, objective functions, further ablation experiments, and more visual results. Specifically, in ablation experiments, we include ablation results on the real-world URHI dataset to validate the effectiveness of the Code-Critic in Sec. 3.1. Besides, throughout the paper, we typically set $T = 8$. To justify this setting, we discuss the effect of the number of iterations in Sec. 3.2.

1. Network Architectures

The structural details of IPC-Dehaze are shown in Tab. 1. In VQGAN, we use a network structure similar to FeMaSR [1] and set the codebook size K to 1024 and embedding dim to 256. To achieve code matching and code evaluation at different resolutions, we use RSTB [11] as the backbone and add a linear projection layer.

2. Objective Functions

2.1. Pretrain: VQGAN Training

Following VQGAN, we adopt pixel-level reconstruction loss $\mathcal{L}_1$ and code-level loss $\mathcal{L}_{code}$ to train the Encoder E_H, Decoder D_H, and Codebook C:

$$\mathcal{L}_1 = \|I_h - I_{rec}\|_1, \tag{1}$$

$$\begin{aligned}\mathcal{L}_{code} = \|sg(z_c) - Z_h^{(i,j)}\|_2^2 + \beta\|z_c - sg(Z_h^{(i,j)})\|_2^2 \\ + \lambda_g\|\text{CONV}(Z_h^{(i,j)}) - \Phi(I_h)\|_2^2,\end{aligned} \tag{2}$$

where $sg(\cdot)$ is the stop-gradient operation, $\beta = 0.25$ and $\lambda_g = 0.1$ in this training. The CONV$(\cdot)$ and $\Phi(\cdot)$ denote a convolution layer and a pre-trained VGG19 [13], respectively.

To restore better texture, we use perceptual loss $\mathcal{L}_{per}$ [9], and adversarial loss $\mathcal{L}_{adv}$ [5] as part of the loss function:

$$\mathcal{L}_{per} = \|\Phi(I_h) - \Phi(I_{rec})\|_1, \tag{3}$$

$$\mathcal{L}_{adv} = \log D(I_h) + \log(1 - D(I_{rec})). \tag{4}$$

In the pre-training stage, the loss is expressed as:

$$\mathcal{L}_{VQGAN} = \mathcal{L}_1 + \mathcal{L}_{code} + \mathcal{L}_{per} + \lambda_{adv}\mathcal{L}_{adv}, \tag{5}$$

where $\lambda_{adv} = 0.1$.

2.2. Stage I: Code-Predictor Training

In this training stage, we use $\mathcal{L}_1$, $\mathcal{L}_{per}$, and $\mathcal{L}_{adv}$ to train the Encoder E_L. In addition, we use the cross-entropy loss $\mathcal{L}_\theta$ to train the Code-Predictor. The total loss is expressed as:

$$\mathcal{L}_\theta = -\sum_{i=0}^{N} S_h^{(i)} \log p_\theta(S^{(i)}|Z_t), \tag{6}$$

$$\mathcal{L}_{total} = \mathcal{L}_1 + \mathcal{L}_{per} + \lambda_{adv}\mathcal{L}_{adv} + \mathcal{L}_\theta, \tag{7}$$

where $\lambda_{adv} = 0.1$, S_h is the code sequence from the clean image, and S is the code sequence predicted by Code-Predictor. Z_t denotes the fused tokens and p_θ denotes the output distribution of Code-Predictor.

2.3. Stage II: Code-Critic Training

In the training stage II, we keep all other modules fixed, exclusively train the Code-Critic, and utilize only binary cross-entropy loss:

$$\mathcal{L}_\phi = -\sum_{i=0}^{N} M^{(i)} \log p_\phi(S^{(i)}) + (1 - M^{(i)}) \log(1 - p_\phi(S^{(i)})), \tag{8}$$

*Corresponding author

Layers	Configuration	Output Size
Conv_in	$c_{in}=3\ c_{out}=64\ ksz=4$	$(h,w,64)$
Block1	$c_{in}=64\ c_{out}=128\ ksz=3\ stride=2$	$(h/2,w/2,128)$
Block2	$c_{in}=128\ c_{out}=256\ ksz=3\ stride=2$	$(h/4,w/4,256)$
Before_quant	$c_{in}=256\ c_{out}=256\ ksz=1\ stride=1$	$(h/4,w/4,256)$
RSTB_block1	$\begin{bmatrix} ws=8 \\ d=256 \\ head=8 \\ depth=6 \end{bmatrix} \times 4$	$(h/4,w/4,256)$
Norm1	d=256	$(h/4,w/4,256)$
Linear1	$f_{in}=256\ f_{out}=1024$	$(h/4,w/4,1024)$
RSTB_block2	$\begin{bmatrix} ws=8 \\ d=256 \\ head=8 \\ depth=6 \end{bmatrix} \times 2$	$(h/4,w/4,256)$
Norm2	d=256	$(h/4,w/4,256)$
Linear2	$f_{in}=256\ f_{out}=1$	$(h/4,w/4,1)$
Codebook	$K=1024\ d=256$	$(h/4,w/4,256)$
After_quant	$c_{in}=256\ c_{out}=256\ ksz=3\ stride=1$	$(h/4,w/4,256)$
Upsample	ratio=2	$(h/2,w/2,256)$
Block3	$c_{in}=256\ c_{out}=128\ ksz=3\ stride=1$	$(h/2,w/2,128)$
SFT_block1	$c_{in}=128\ c_{out}=128\ ksz=3\ stride=1$	$(h/2,w/2,128)$
Upsample	ratio=2	$(h,w,128)$
Block4	$c_{in}=128\ c_{out}=64\ ksz=3\ stride=1$	$(h,w,64)$
SFT_block2	$c_{in}=64\ c_{out}=64\ ksz=3\ stride=1$	$(h,w,64)$
Conv_out	$c_{in}=64\ c_{out}=3\ ksz=3\ stride=1$	$(h,w,3)$

Table 1. Architecture details of the IPC-Dehaze. Blue, green, brown, and yellow represent the layers of VQGAN, Code-Predictor, Code-Critic, and SFT, respectively. c_{in}, c_{out}, and ksz are the input channel, output channel, and kernel size, respectively. The ws is the window size and d is the embedding dim. f_{in} is the number of input features and f_{out} is the number of output features. The input of the network is an RGB image $\in \mathbb{R}^{h\times w\times 3}$.

where $M=(S_h \neq S)$ and p_ϕ denotes the output of Code-Critic.

Since the Code-Critic module is only trained to make an accurate evaluation of Code-Predictor's diverse cases, we introduce the sampling temperature when Code-Predictor samples the code sequence S. The partial code is shown in Fig. 1.

3. Ablation Experiments

3.1. Effectiveness of Code-Critic

To further discuss the necessity of Code-Critic, we compare the sampling method based on code confidence and that based on Code-Critic. We conduct quantitative experience on two real-world datasets RTTS and URHI [10], which both contain over 4,000 images. We present the results in Tab. 2.

```
#Fuse the hq_feats and lq_feats with mask
input_feats=hq_feats*~mask+lq_feats*mask
# Get the logits with [b, h*w, K].
logits = net_predictor.transformer(input_feats)
# Add sampling temperature
logits/=Tem
probs = F.softmax(logits, -1)
# Samples the id.
sampled_ids = torch.multinomial(probs), 1)
# Evaluate the sampled_ids
masked_logits = net_critic(sampled_ids,h,w)
```

Figure 1. Partial code for Code-Critic Trainin.

Table 2. Quantitative ablation analysis of the Code-Predictor. Red indicates the best results. In this experiment, we set $T=8$.

(a) Results on RTTS.

Method	MUSIQ↑	PI↓	MANIQA↑	CLIPIQA↑	Q-Align↑	TOPIQ↑
NN Matching Based	58.19	3.25	0.303	0.391	3.25	0.458
Ours (w/o Code-Critic)	57.74	3.32	0.303	0.412	3.36	0.462
Ours	59.60	3.22	0.327	0.44	3.49	0.500

(b) Results on URHI.

Method	MUSIQ↑	PI↓	MANIQA↑	Q-Align↑
NN Matching Based	62.06	3.01	0.350	3.48
Ours (w/o Code-Critic)	60.51	3.13	0.343	3.55
Ours	62.50	3.08	0.364	3.70

3.2. Iteration Number

We investigate the impact of varying the iteration number T on inference performance, as summarized in Tab. 3. To balance performance and efficiency, we set $T=8$ as the default. As shown in Tab. 3, reducing T slightly degrades performance but still outperforms other methods. For faster inference, a smaller T can be selected without the need for network retraining.

Table 3. Quantitative analysis of the different iteration numbers. Red indicates the best results.

T	MUSIQ↑	PI↓	MANIQA↑	CLIPIQA↑	Q-Align↑	TOPIQ↑
3	58.40	3.33	0.314	0.425	3.43	0.478
4	58.71	3.31	0.318	0.431	3.45	0.484
6	58.97	3.30	0.321	0.437	3.47	0.490
8	59.60	3.22	0.327	0.444	3.489	0.500
10	59.23	3.28	0.325	0.442	3.488	0.496

4. Visual Results

4.1. Visualisation of Iterative Decoding

Fig. 2 visualizes the iterative process, and it can be seen that during the iterative process, our results get better and better in terms of dehazing, image quality, and visual effects.

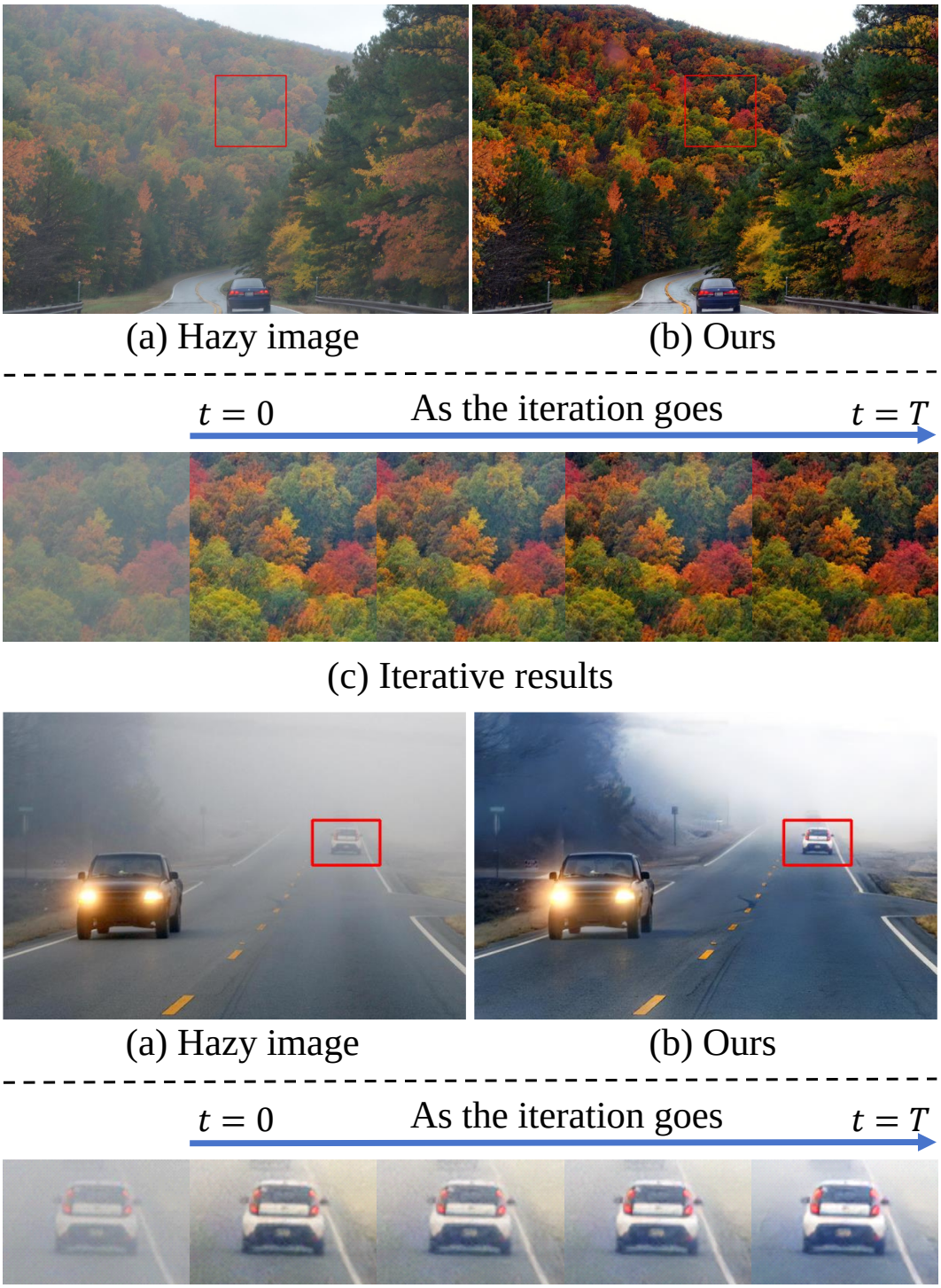

Figure 2. The top-left shows the input image, while the top-right displays our result. Below, the images from left to right depict the intermediate results of the iterative decoding process when $T = 8$.

4.2. More Visual Comparisons

Figs. 3 to 5 show visual comparisons on the RTTS, URHI [10] and Fattal [6] datasets. We compare our method with the methods that have achieved outstanding results in benchmarks: MSDBN [4], Dehamer [8], and DEA-Net [3] as well as real-world image dehazing methods: DAD [12], PSD [2], D4 [15], RIDCP [14], and KA-Net [7]. The results show that our method can achieve more natural, high-quality images, especially in dense hazy areas, which is a significant improvement compared to other methods.

4.3. Failure Case

In this part, we discuss the failure case of our method. To be specific, our method exhibits a progressive approach to dehazing, processing denser haze regions first and gradually moving to thinner regions during iterative dehazing. This gives us a distinct advantage in handling depth-continuous scenes, as high-quality codes from earlier iterations can serve as cues for the Code-Predictor to refine subsequent predictions. However, the performance is limited in depth-discontinuous scenes, as illustrated in Fig. 6. While our method performs exceptionally well in depth-continuous regions (outside the red box), it struggles as the other methods in depth-discontinuous regions (inside the red box, such as trees or rocks). Such depth-discontinuous scenarios present significant challenges for all existing dehazing methods.

References

[1] Chaofeng Chen, Xinyu Shi, Yipeng Qin, Xiaoming Li, Xiaoguang Han, Tao Yang, and Shihui Guo. Real-world blind super-resolution via feature matching with implicit high-resolution priors. In *ACM MM*, pages 1329–1338, 2022. 1

[2] Zeyuan Chen, Yangchao Wang, Yang Yang, and Dong Liu. Psd: Principled synthetic-to-real dehazing guided by physical priors. In *CVPR*, pages 7180–7189, 2021. 4

[3] Zixuan Chen, Zewei He, and Zhe-Ming Lu. Dea-net: Single image dehazing based on detail-enhanced convolution and content-guided attention. *IEEE TIP*, 33:1002–1015, 2024. 4

[4] Hang Dong, Jinshan Pan, Lei Xiang, Zhe Hu, Xinyi Zhang, Fei Wang, and Ming-Hsuan Yang. Multi-scale boosted dehazing network with dense feature fusion. In *CVPR*, 2020. 4

[5] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, pages 12873–12883, 2021. 1

[6] Raanan Fattal. Dehazing using color-lines. *ACM TOG*, page 1–14, 2014. 4

[7] Yuxin Feng, Long Ma, Xiaozhe Meng, Fan Zhou, Risheng Liu, and Zhuo Su. Advancing real-world image dehazing: Perspective, modules, and training. *IEEE TPAMI*, 46(12): 9303–9320, 2024. 4

[8] Chun-Le Guo, Qixin Yan, Saeed Anwar, Runmin Cong, Wenqi Ren, and Chongyi Li. Image dehazing transformer with transmission-aware 3d position embedding. In *CVPR*, pages 5812–5820, 2022. 4

[9] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016. 1

[10] Boyi Li, Wenqi Ren, Dengpan Fu, Dacheng Tao, Dan Feng, Wenjun Zeng, and Zhangyang Wang. Benchmarking single-image dehazing and beyond. *IEEE TIP*, page 492–505, 2019. 2, 4

[11] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *ECCV*, pages 1833–1844, 2021. 1

[12] Yuanjie Shao, Lerenhan Li, Wenqi Ren, Changxin Gao, and Nong Sang. Domain adaptation for image dehazing. In *CVPR*, 2020. 4

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1

[14] Rui-Qi Wu, Zheng-Peng Duan, Chun-Le Guo, Zhi Chai, and Chongyi Li. Ridcp: Revitalizing real image dehazing via high-quality codebook priors. In *CVPR*, pages 22282–22291, 2023. 4

[15] Yang Yang, Chaoyue Wang, Risheng Liu, Lin Zhang, Xiaojie Guo, and Dacheng Tao. Self-augmented unpaired image dehazing via density and depth decomposition. In *CVPR*, pages 2037–2046, 2022. 4

Figure 3. Visual comparison on RTTS. **Zoom in for best view**.

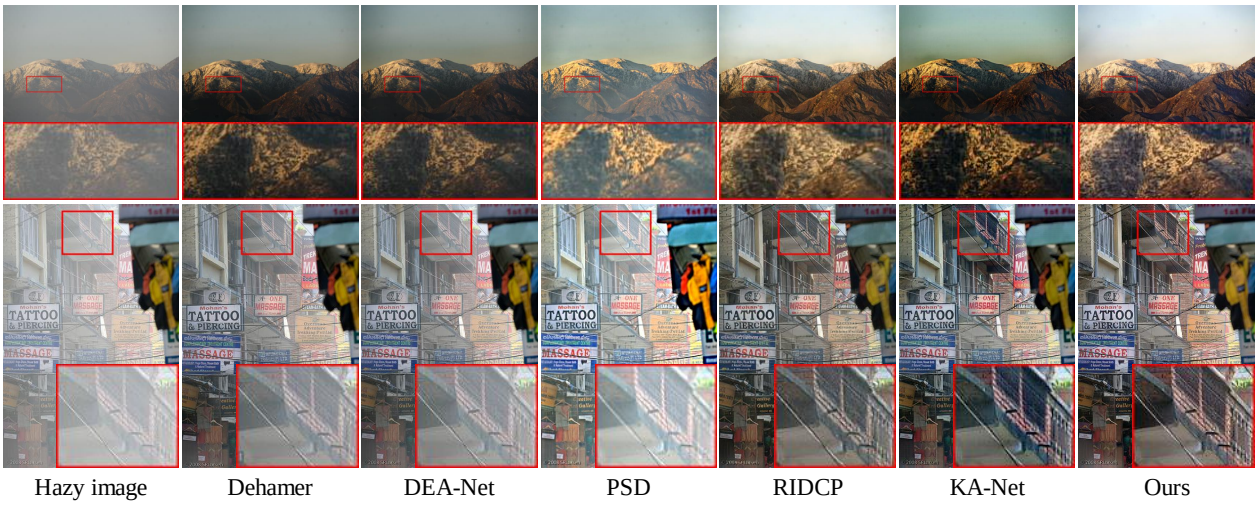

Figure 4. Visual comparison on Fattal. **Zoom in for best view**.

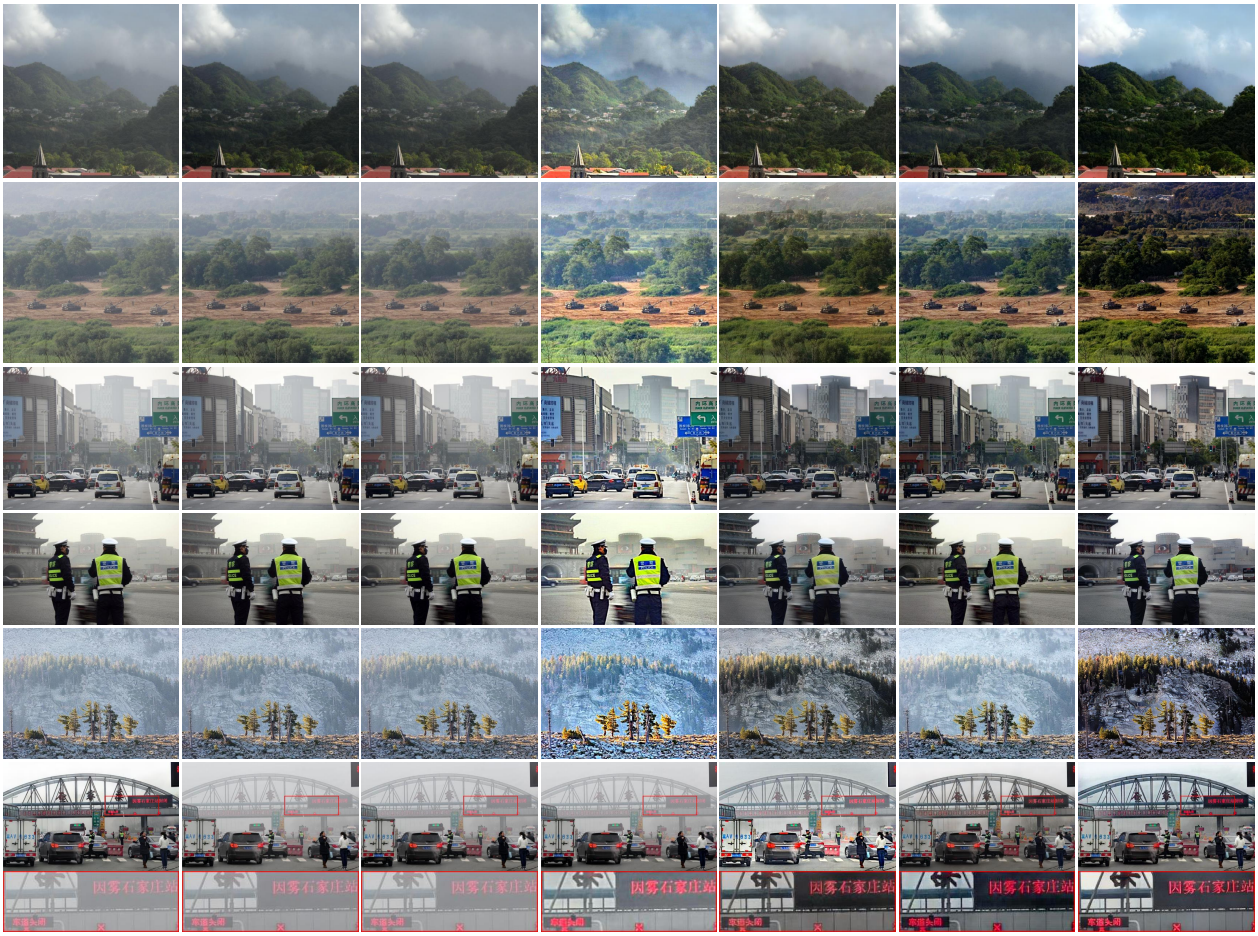

Figure 5. Visual comparison on URHI. **Zoom in for best view**.

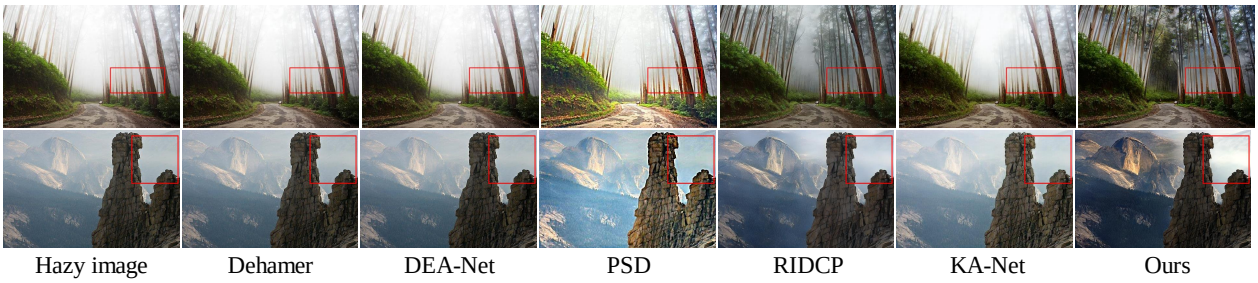

Figure 6. Failure Case. As shown in the red boxes, depth-discontinuous regions present significant challenges for existing dehazing methods. While our method performs well in the depth-continuous regions outside the red box, it struggles inside the red box, where trees and rocks disrupt depth continuity, rendering our method as ineffective as other approaches.