# Supplementary Material for DCCF: Deep Comprehensible Color Filter Learning Framework for High-Resolution Image Harmonization

Ben Xue[1⋆], Shenghui Ran[2], Quan Chen[2⋆⋆],
Rongfei Jia[2], Binqiang Zhao[2], and Xing Tang[2]

[1] Academy for Advanced Interdisciplinary Studies, Peking University, China
[2] Alibaba Group, China

## 1 Introduction

For better acknowledgement of details, we provide several supplementary sections to demonstrate the mechanism and capability of DCCF.

In Section 2 and 4, we provide more experiment results on high-resolution images along with the efficiency comparison. In Section 5, we further show the low-resolution results to demonstrate the efficacy of designed filters.

In Section 7.1, we show that a constrained rotation matrix in RGB space can actually rotate the color angle in HSV space. Along with the disentanglement in *high resolution assembly module* in Section 7.2, we also show that DCCF like $F_{hue}$ can learn unconstrained rotation matrix which can be projected on its current HSV plane to directly affect the specific HSV channel.

In Section 8, we demonstrate the numerical procedure of standard and smoothing(ours) HSV supervision strategies.

In Section 9, we show the capability of DCCF to adjust specific image attribute. Each filter of this family can be picked out to qualify its sub-task. In Section 10 and Section 11, we provide more visualization results of comprehensible interaction and high-resolution results. In Section 12, we discuss the potential limitation of our framework.

## 2 High-Resolution Results with Different Backbones

Note that our DCCF can be plugged into different backbones, we select DIH [11] and $S^2$AM [4] for experiment. [11] shares similary U-net[8] architecture with iDIH-HRNet [9] except for the extra pretrained visual feature from HRNet [10]. [4] uses spatial-separated attention module in decoder to aggregate semantic information. For fair comparison, we keep learning strategy the same as DCCF

⋆ Finish this work during an internship at Alibaba Group. Email: xueben@pku.edu.cn.
⋆⋆ Corresponding author. Email: myctllmail@163.com.

Table 1: **Quantative results** on the iHarmony4 original-resolution test sets with other backbones. '-' means not able to obtain results due to memory limitation. 'DCCF-*' means DCCF filters backboned by *.

| | Entire Dataset | | HCOCO | | HAdobe5k | | HFlickr | | Hday2night | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ |
| DIH [11] | - | - | 36.39 | 36.56 | - | - | 186.38 | 30.78 | 61.89 | 35.40 |
| DIH [11] + BU | 51.28 | 34.39 | 39.35 | 34.92 | 44.99 | 34.81 | 129.48 | 30.14 | 51.07 | 36.77 |
| DIH [11] + GF[7] | 43.10 | 35.35 | 30.73 | 36.11 | 41.57 | 35.34 | 109.99 | 31.13 | 50.00 | 37.09 |
| DIH [11] + BGU[1] | 34.37 | 36.47 | 23.16 | 37.21 | 34.11 | 36.63 | 90.24 | 32.18 | 51.78 | 36.64 |
| DCCF-DIH | **33.39** | **36.87** | **21.60** | **37.81** | **34.09** | **36.77** | **89.86** | **32.24** | **49.93** | **37.23** |
| $S^2$AM [4] | - | - | 33.43 | 36.93 | - | - | 186.70 | 30.78 | 57.48 | 36.05 |
| $S^2$AM [4] + BU | 44.02 | 35.02 | 36.36 | 35.30 | 33.01 | 35.95 | 112.74 | 30.77 | 41.84 | 37.44 |
| $S^2$AM [4] + GF[7] | 35.88 | 36.05 | 27.83 | 36.53 | 29.63 | 36.58 | 93.06 | 31.85 | **40.94** | **37.79** |
| $S^2$AM [4] + BGU[1] | 27.94 | 37.18 | 20.25 | 37.73 | 24.71 | **37.68** | 73.82 | 32.99 | 42.49 | 37.28 |
| DCCF-$S^2$AM | **26.74** | **37.59** | **18.41** | **38.43** | **24.39** | 37.65 | **73.18** | **33.12** | 44.25 | 37.36 |

when training these backbones, where we use RandomResizedCrop and RandomHorizontalFlip as data augmentation, foreground-normlized MSE [9] as training loss, XavierGluon (gaussian, magnitute=0.2) as weights initializer. We use bilinear upsamplng (BU), guided filter (GF [7]) and bilateral grid upsampling (BGU [1]) as post-processing methods. Experiments in Table 1 show that DCCF constantly outputs these baselines which demonstrates the robustness and generality of our framework.

## 3 Comparison with CDTNet

We finetune our model under weaker resolution settings (1024×1024, 2048×2048) on HAdobe5k subset to compare with the recent high resolution harmonization method CDTNet [2]. To make fair comparison, we use the same backbone $S^2$AM-256 with [2] (i.e CDTNet-256). The result is shown in Table 2. Our DCCF has better performance on higher resolution setting (2048×2048) than CDTNet-256. It is also observed that the performance of [2] drops significantly as resolution increases, while our method maintains stable performance. Note that other high resolution experiments in our paper are conducted under a much stronger setting: the original resolution of HAdobe5k can range up to 6048×4032.

Table 2: **Quantative comparison** with CDTNet [2] on HAdobe5k subset.

| Resolution | Method | MSE↓ | PSNR↑ | fMSE↓ | SSIM↑ |
|---|---|---|---|---|---|
| 1024 | CDTNet-256 | 21.24 | **38.77** | **152.13** | **0.9868** |
| ×1024 | DCCF | **21.12** | 38.38 | 171.17 | 0.9852 |
| 2048 | CDTNet-256 | 29.02 | 37.66 | 198.85 | 0.9845 |
| ×2048 | DCCF | **21.35** | **38.47** | **174.78** | **0.9856** |

Table 3: **Efficiency comparison** between DCCF and advanced post-processing methods on different resolutions from 1024 to 3072. 'T-C' represents cpu time (ms), 'T-G' represents gpu time (ms) and 'Mem' represents memory usage (MB). Note that '-' means no offical implementation is found.

| | 1024 × 1024 | | | 2048 × 2048 | | | 3072 × 3072 | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | T-C (ms) ↓ | T-G (ms) ↓ | Mem (MB) ↓ | T-C (ms) ↓ | T-G (ms) ↓ | Mem (MB) ↓ | T-C (ms) ↓ | T-G (ms) ↓ | Mem (MB) ↓ |
| iDIH-HRNet[9] | 420 | 231 | 1641 | 41040 | 907 | 4233 | 139768 | 2042 | 8551 |
| iDIH-HRNet[9] + GF[7] | 642 | 80.2 | 983 | 2001 | 160 | 1513 | 10181 | 391 | 2483 |
| iDIH-HRNet[9] + BGU[1] | 9932 | - | 2893 | 20803 | - | 4042 | 29836 | - | 8173 |
| DCCF-iDIH-HRNet | 762 | 104 | 1259 | 3289 | 286 | 2607 | 6517 | 545 | 4845 |

## 4 Efficiency Comparison

To compare efficiency between DCCF and existed post-processing upsampling methods, we test these methods on different resolutions from 1024×1024 to 3072×3072. The experiment is conducted on a x86-64 machine (72 cores, ubuntu 18.04) with a 12GB Nivida Titan X gpu card. We test cpu time (T-C), gpu time (T-G) and memory usage (Mem) for evaluation metrics. Each method is warmed up by 10 times and averaged by another 20 times forward passes. As for detailed parameters which could influence efficiency metrics, GF [7] uses $r = 8$ kernel size, BGU [1] uses default $16 \times 16 \times 3 \times 4$ grid size.

Experiments in Table 3 show that GF [7] take most of the leads in efficiency metrics, that is mainly because it only involves several basic box filters, however it losses too much performance compared to DCCF (MSE/PSNR, 24.65/37.87→35.47/36.00). BGU [1] explicitly estimates bilateral grids which contain image-to-image transformation coefficients, therefore the performance is higher compared to GF [7], however the efficiency drops far behind since it needs extra optimization procedure. To this end, DCCF achieves a good trade-off between performance and efficiency.

## 5 Low-Resolution Results

To further show the efficacy of our designed comprehensible filters, we compare our approach with other state-of-the-art deep models [27,6,5,9] on the iHarmony4 low-resolution ($256 \times 256$) test sets. The results are shown in Table 4. It is interesting that our approach also outperforms the previous best one [23] slightly on the entire iHarmony4 dataset on low-resolution, which may due to the appropriate design of filters and extra supervision from auxiliary HSV losses via our framework.

## 6 Ablation Study on Operation Order

According to our survey, many designers and artists in Photoshop community tend to harmonize an image in the order of 'value, saturation, hue'. We regard this phenomenon as a common convention thus design our framework in such an order. However we think the operation order of DCCF filter is also meaningful

Table 4: **Quantative results** on the iHarmony4 low-resolution test sets.

| | Entire Dataset | | HCOCO | | HAdobe5k | | HFlickr | | Hday2night | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ |
| DIH [11] | 76.77 | 33.41 | 51.85 | 34.69 | 92.65 | 32.28 | 163.38 | 29.55 | 82.34 | 34.62 |
| $S^2$AM [4] | 59.67 | 34.35 | 41.07 | 35.47 | 63.40 | 33.77 | 143.45 | 30.03 | 76.61 | 34.50 |
| DoveNet [3] | 52.36 | 34.75 | 36.72 | 35.83 | 52.32 | 34.34 | 133.14 | 30.21 | 54.05 | 35.18 |
| IntrinsicIH [5] | 38.71 | 35.90 | 24.92 | 37.16 | 43.02 | 35.20 | 105.13 | 31.34 | 55.53 | 35.96 |
| iDIH-HRNet [9] | 22.81 | 38.18 | **14.35** | **39.53** | 23.43 | 37.18 | 61.42 | 33.84 | **45.09** | **38.08** |
| DCCF | **22.05** | **38.50** | 14.87 | 39.52 | **19.90** | **38.27** | **60.41** | **33.94** | 49.32 | 37.88 |

thus we make corresponding ablation experiment. We investigate the whole 3! combinations 'VSH', 'HVS', 'SHV', 'VHS', 'HSV', 'SVH' on low resolution images in Table 5. Note that the results are slightly different with Table 4 because Table 5 uses fewer training epochs, but we ensure that all abaltions in Table 5 share the same parameter setting. A tentative conclusion is that the order would affect final results and different subsets also require different optimal operation orders. A possible solution is introducing re-enforcement learning (RL) to decide which is the best operation order when processing a certain given image. This may inspire our future works.

Table 5: **Ablation study** of different orders.

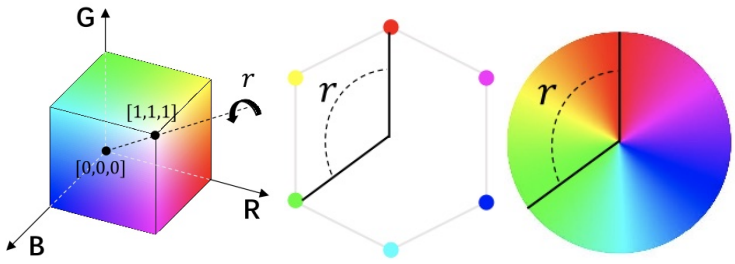
| | Entire Dataset | | HCOCO | | HAdobe5k | | HFlickr | | Hday2night | |
|---|---|---|---|---|---|---|---|---|---|---|
| Operation Order | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ | MSE ↓ | PSNR ↑ |
| V→S→H (default) | 22.52 | 38.57 | 14.20 | **39.73** | 22.54 | 38.12 | 61.80 | 33.92 | 45.54 | 37.51 |
| H→V→S | 22.90 | 38.53 | 14.45 | 39.66 | 23.66 | 38.12 | 60.32 | 33.93 | 49.78 | 37.67 |
| S→H→V | 22.88 | 38.43 | 14.90 | 39.48 | 21.81 | 38.17 | 63.94 | 33.74 | **41.64** | **37.94** |
| V→H→S | **22.11** | **38.63** | **14.02** | 39.71 | 21.98 | 38.35 | **59.54** | **33.98** | 51.77 | 37.58 |
| H→S→V | 22.45 | 38.57 | 14.67 | 39.57 | **21.29** | **38.45** | 61.97 | 33.87 | 45.78 | 37.69 |
| S→V→H | 22.94 | 38.53 | 14.25 | 39.69 | 23.08 | 38.06 | 61.77 | 33.90 | 59.10 | 37.33 |

## 7 Hue Filter and Disentanglement

### 7.1 Hue Filter

Let $\boldsymbol{x} : (x_r, x_g, x_b)$ indicates the RGB values for one pixel in image, $\boldsymbol{z} : (z_h, z_s, z_l)$ is the corresponding point in HSV space. Let $\boldsymbol{\Delta}$ is a learnable 3x4 affine transformation matrix in RGB space that contains a rotation matrix $\boldsymbol{R}$ and translation vector $t$, and $\boldsymbol{r}$ is a radian moving on the hue ring in HSV space.

According to the theory in [6], to rotate the hue by $\boldsymbol{r}$, we perform a 3D rotation of RGB colors about the diagonal vector [1.0 1.0 1.0] as as illustrated in Fig. 1. The resulting matrix will rotate the hue of the input RGB colors. A rotation of $2\pi/3$ will exactly map Red into Green, Green into Blue and Blue into Red. The matrix processing in [6] makes an approximation that the diagonal axis in RGB space is equivalent to the hue axis in HSV space. This transformation




(a) 3D Rotation (b) Diagonal (c) Hue

Fig. 1: Illustration of rotation matrix. Viewing 3D RGB cube model (a) from the diagonal perspective, we can get the hexagon in (b), which is an approximation of real color circle in hue space (c). Therefore rotation $\boldsymbol{r}$ is equivalent in above three models.

has one problem, however, the luminance of the input colors is not preserved. This can be fixed by shearing the value plane to make it horizontal.

We suppose that one could find a suitable rotation matrix $\boldsymbol{R}$ in RGB color space that is equivalent to [6]. Therefore, it is possible to learn an affine color transformation function $f_{hue}(\boldsymbol{x}; \boldsymbol{\Delta})$ in RGB color space, which contains a rotation function $\boldsymbol{R}$ that could be the parameters for the corresponding hue rotation function $f_{hue}(h; \boldsymbol{r})$ in HSV space. Exactly, $f_{hue}(h; \boldsymbol{r})$ is the desired linear transformation for hue filter $F_{hue}$. It is obvious that the linear transformation $\boldsymbol{\Delta}$ could map one RGB point $\boldsymbol{x_1}$ to any other RGB point $\boldsymbol{x_2}$, and the corresponding HSV point $\boldsymbol{z_1}$ moves to $\boldsymbol{z_2}$. To avoid the modification along L and S axis, we perform HSV disentangle by projecting the path $(\boldsymbol{z_1} \rightarrow \boldsymbol{z_2})$ on H plane to get $\boldsymbol{z}_{2\|h}$.

### 7.2 Effect of HSV Disentanglement

(a) Input (b) $F_{hue}$ on RGB (c) Disentangle

Fig. 2: Illustration of disentanglement. *High resolution assembly module* extracts the H channel of output and concatenates it with the input's V and S. This operation prevents $F_{hue}$ from changing V and S.

We show the importance of *high resolution assembly module* disentanglement in Fig. 2. Taking $F_{hue}$ as example, naively applying it in RGB space actually

changes V and S simultaneously. DE (Disentangle) avoids this situation by extracting the H channel of output and concatenating it with the input's V and S. This ensures the action of $F_{hue}$ won't corrupt the result of $F_{val}$ and $F_{sat}$.

Note that this disentanglement strategy is equivalent to the projection action in Section 7.1. We only need the path ($\boldsymbol{z_1} \rightarrow \boldsymbol{z_2}$) on the H plane: ($\boldsymbol{z}_{1\|h} \rightarrow \boldsymbol{z}_{2\|h}$). Similarly, applying $F_{val}$ + DE and $F_{sat}$ + DE will generate projection path on V plane and S plane, which are ($\boldsymbol{z}_{1\|l} \rightarrow \boldsymbol{z}_{2\|l}$), ($\boldsymbol{z}_{1\|s} \rightarrow \boldsymbol{z}_{2\|s}$). The orthogonality of HSV space will ensure that these paths won't cross each other.

## 8 Auxiliary HSV Loss

### 8.1 Standard HSV Decomposition

The numerical conversion of HSV and RGB value is performed as:

$$V = C_{max} \tag{1}$$

$$S = \frac{C_{max} - C_{min}}{C_{max}} \tag{2}$$

$$H = \begin{cases} \pi/6 \times (\frac{G-B}{C_{max}-C_{min}} mod 6), C_{max} = R \\ \pi/6 \times (\frac{B-R}{C_{max}-C_{min}} + 2), C_{max} = G \\ \pi/6 \times (\frac{R-G}{C_{max}-C_{min}} + 4), C_{max} = B \end{cases} \tag{3}$$

This expression tends to result in noise points because it is based on numerical values rather than physical characteristics.

### 8.2 Smoothing V

The generation of $V_{smooth}$ is straightforward. We apply gaussian blur on the original $V$ decomposition to get $V_{smooth}$, where we set variance scale $std = 1.5$ and kernel size $K = 5$ in implementation.

### 8.3 Smoothing S

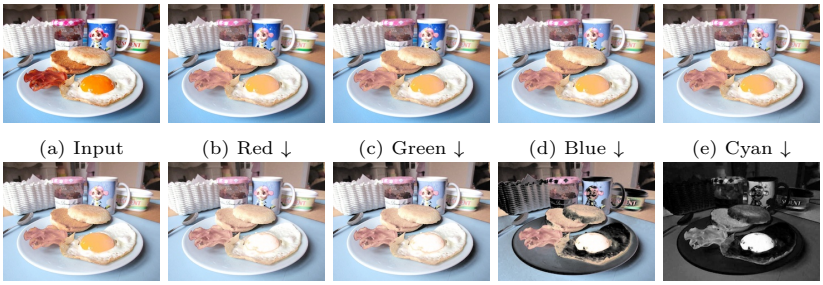
Different from Eq. 2, we follow the operation in photoshop to get a smooth map. First, we perform a selective color adjustment by setting all the colored tunes to -100%: red, yellow, green, cyan, blue, and magenta(RGB, CMY). Then for the blacks, whites and neutrals(BWN), we enhance them to 100%. Noted that the param used here ranges from -100% to 100%, actually it is exactly the same $\sigma$ we used in our saturation filter. The detailed procedure is shown in Fig. 3. The numerical expression can be expressed as:

$$S_t = F_{sat}(S_{t-1}; \sigma_t) * ROI(c_t), t = 1, 2, ...9 \tag{4}$$

$$S_{smooth} = I_9, S_0 = I_1 = F_{val}(I) \tag{5}$$

Where $c_t$=[R,G,B(Blue),C,M,Y,B(Black),W,N], $\sigma_t = -1$ for $t \in$ [1,2,3,4,5,6], and $\sigma_t = 1$ for $t \in$ [7,8,9]. Note that $ROI(c_t)$ is the regions within this color.

The result is a color map that shows you saturation levels across the scene. Darker shades of gray are less saturated, and lighter shades are more saturated.



(a) Input (b) Red ↓ (c) Green ↓ (d) Blue ↓ (e) Cyan ↓

(f) Megenta ↓ (g) Yellow ↓ (h) Black ↑ (i) White ↑ (j) Neutral ↑

Fig. 3: Procedure of smoothing saturation map. It is arranged from left to right, top to bottom. Note that (j) is also the final result of our smoothing S.

### 8.4 Smoothing H

As for $H_{smooth}$, we first convert RGB image into HSV space and set the V channel to 0.8, S channel to 0.5, then convert it back to RGB space.

## 9 Intermediate Result Visualizations

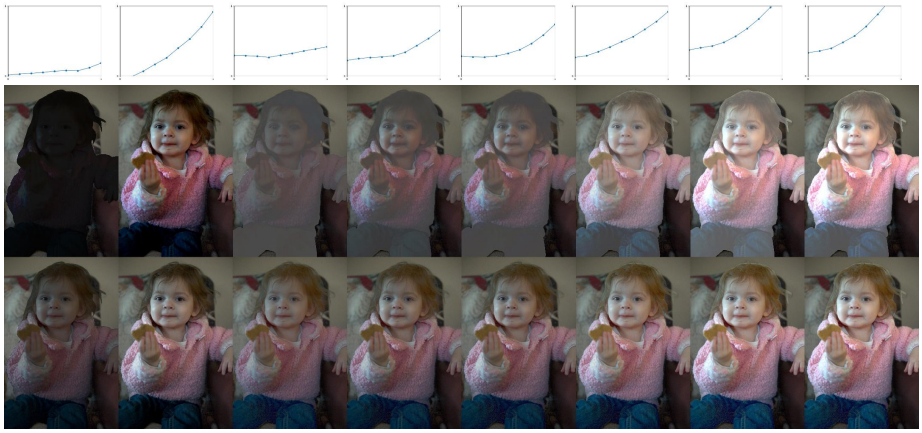
We show that our intermediate result of DICF matches their design purpose in Fig. 6. $I_c$ is the input image. $I_1, I_2, I_3, I_4$ are outputs after $F_{val}, F_{sat}, F_{hue}, F_{attn}$. $H_c, S_c, V_c$ are input's HSV maps. $H_3, S_2, V_1$ are their harmonized counterparts. $I_{gt}, V_{gt}, S_{gt}, H_{gt}$ are ground truths.

As illustrated in the upper part of Fig. 6, each intermediate result of $I_1 = F_{val}(I_c)$, $I_2 = F_{sat}(I_1)$, $I_3 = F_{hue}(I_2)$ not only changes its corresponding image attributes: *value*, *saturation* and *hue*, but also maintains reasonable visual quality. This is untrivial since we didn't apply direct RGB loss on $I_1$,$I_2$,$I_3$, instead we only apply auxiliary HSV loss on its specific channels. This means that users can choose any of $I_1, I_2, I_3, I_4$ as their desired output if they only want to change part of these attributes, while previous works only provides a final result which is $I_4$ in our framework. This brings more flexility and robustness for users when approaching DCCF.

We further illustrate the comprehensibility in the bottom part of Fig. 6, the modified channel is closer to ground truth after the operation of DCCF. This also proves the effect of HSV loss in our framework.

## 10 Comprehensible Interaction

To interact with $F_{val}$, in standard image processing softwares, users need to provide a curve to adjust value. In our framework, users can set $[\phi_0, ...\phi_m]$ to approximate a curve. As shown in Fig. 4(a), the first row is the visualization of several curves provided by user. The second row is the result of directly applying these curves on RGB images which degrades image quality since each pixel



(a) Illustration of interactive value adjustment.

(b) Illustration of interactive saturation adjustment.

Fig. 4: Interactive adjustment. Upper: users' global adjustment. Bottom: interactive adjustment with DCCF.

has the same tuning curve. The third row is the result of applying a weighted fusion($\alpha = 0.5$) of user's global curve and DCCF's filter map $F_{val}$.

To interact with $F_{sat}$, users can change image saturation by $\sigma \in [-1, 1]$. As shown in Fig. 4(b), we set a series of $\sigma \in [-0.8, -0.6, -0.4, -0.2, 0.2, 0.4, 0.6, 0.8]$ to generate the first row, which is a global adjustment and each pixel has the same $\sigma$ and could suffer from over-saturation. The second row is the weighted fusion($\alpha = 0.5$) of user's $\sigma$ and DCCF's filter map $F_{sat}$.

## 11 More High-Resolution Visualizations

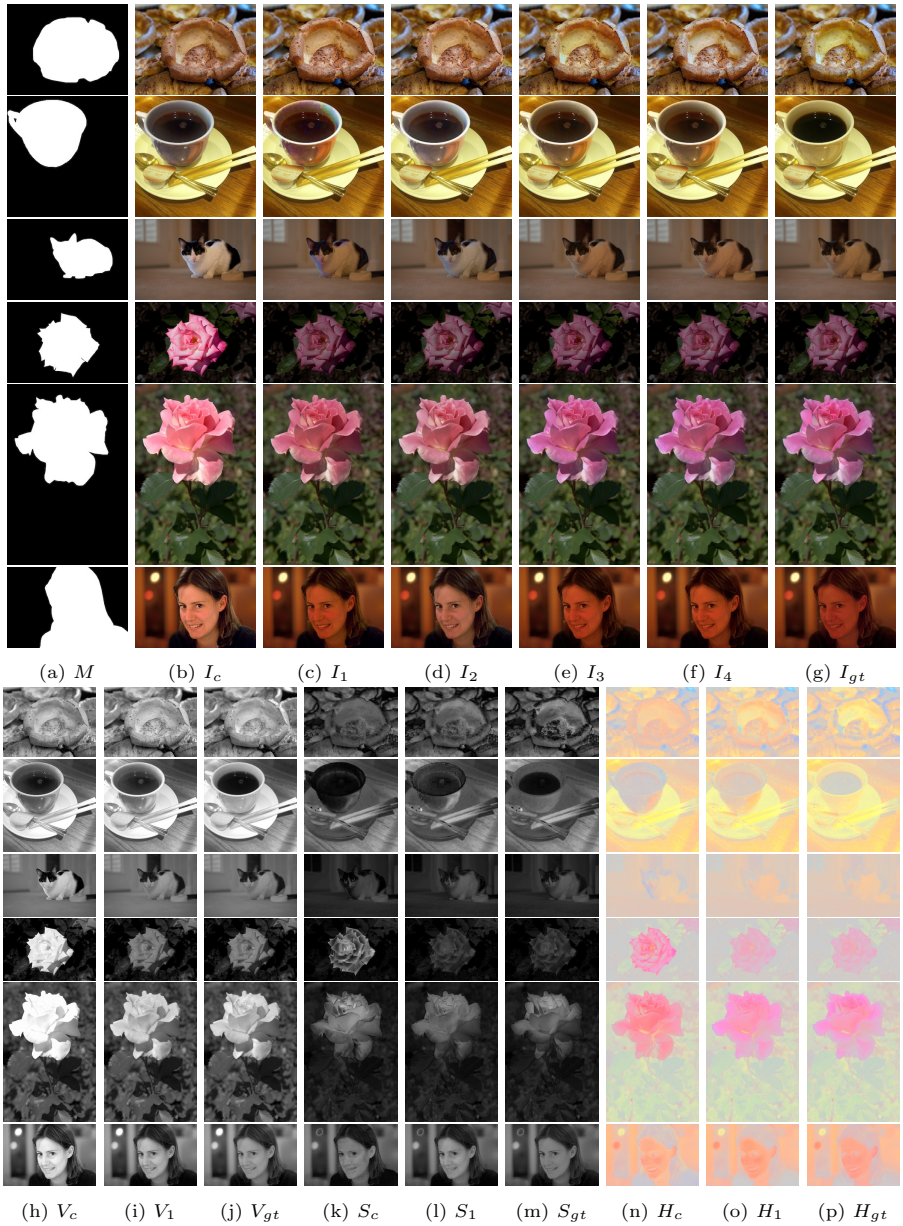
We provide more visualizations of final results compared with previous methods in Fig. 7. Since DCCF is an end-to-end framework, it has strong transformation capability while maintaining high-resolution details.

## 12 Limitations

(a) BGU [1] (b) DCCF (c) GT

Fig. 5: Potential limitation. The second row is amplified details. It is observed that even DCCF learns better color adjustment, the detail starts to blur on high frequency regions like *leaf*.

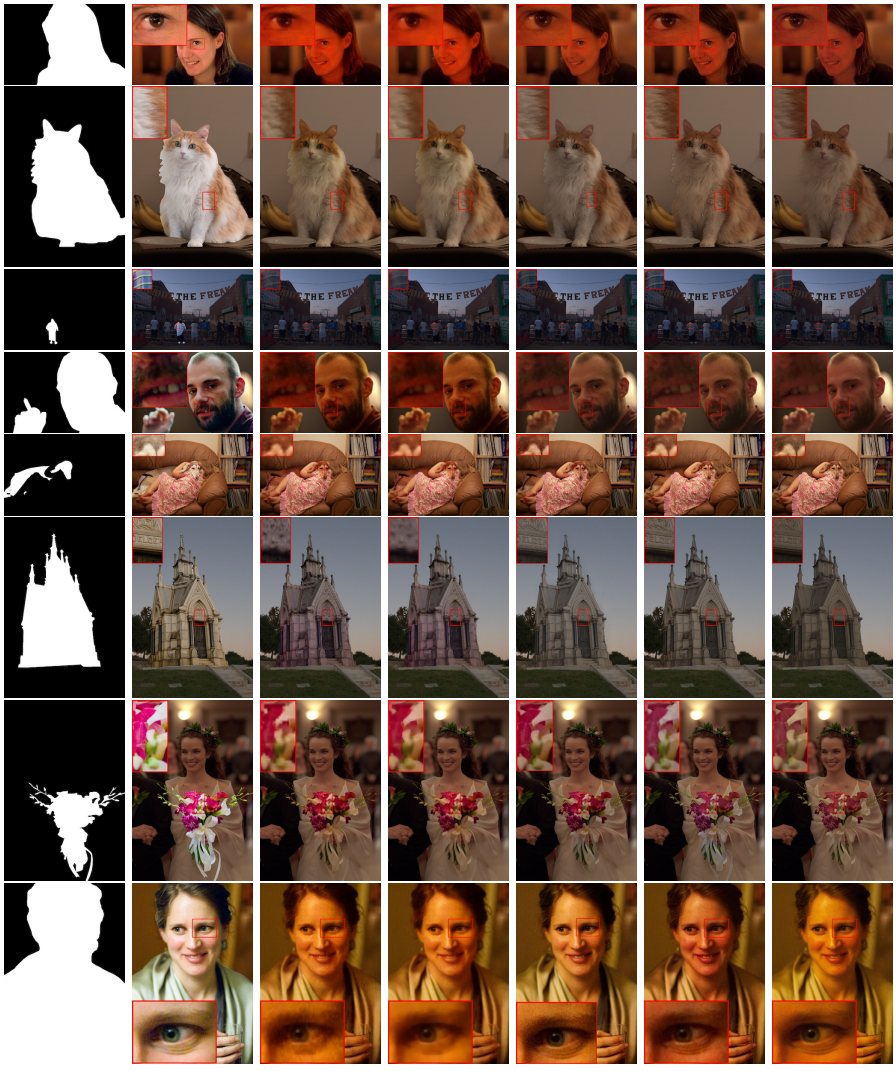
Since the high resolution result is guided by low resolution stream in our framework, the claim of insensitivity to resolution is valid only if the processed image has enough information shared across all signal frequency. In the case of extremely high frequency contents, it may fail to reharmonize images properly, see Fig 5.



(a) $M$ (b) $I_c$ (c) $I_1$ (d) $I_2$ (e) $I_3$ (f) $I_4$ (g) $I_{gt}$

(h) $V_c$ (i) $V_1$ (j) $V_{gt}$ (k) $S_c$ (l) $S_1$ (m) $S_{gt}$ (n) $H_c$ (o) $H_1$ (p) $H_{gt}$

Fig. 6: Intermediate results.



(a) Mask (b) Input (c) BU (d) GF[7] (e) BGU[1] (f) Ours (g) GT

Fig. 7: Visualization of high-resolution results.

## References


1. Chen, J., Adams, A., Wadhwa, N., Hasinoff, S.W.: Bilateral guided upsampling. ACM Transactions on Graphics (TOG) **35**(6), 1–8 (2016)
2. Cong, W., Tao, X., Niu, L., Liang, J., Gao, X., Sun, Q., Zhang, L.: High-resolution image harmonization via collaborative dual transformations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 18470–18479 (2022)
3. Cong, W., Zhang, J., Niu, L., Liu, L., Ling, Z., Li, W., Zhang, L.: Dovenet: Deep image harmonization via domain verification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8394–8403 (2020)
4. Cun, X., Pun, C.M.: Improving the harmony of the composite image by spatial-separated attention module. IEEE Transactions on Image Processing (TIP) **29**, 4759–4771 (2020)
5. Guo, Z., Zheng, H., Jiang, Y., Gu, Z., Zheng, B.: Intrinsic image harmonization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16367–16376 (2021)
6. Haeberli, P.: Matrix operations for image processing. Grafica Obscura website (1993), http://graficaobscura.com/matrix/index.html
7. He, K., Sun, J., Tang, X.: Guided image filtering. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) **35**(6), 1397–1409 (2013)
8. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention (MICCAI). pp. 234–241 (2015)
9. Sofiiuk, K., Popenova, P., Konushin, A.: Foreground-aware semantic representations for image harmonization. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 1620–1629 (2021)
10. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
11. Tsai, Y.H., Shen, X., Lin, Z., Sunkavalli, K., Lu, X., Yang, M.H.: Deep image harmonization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3789–3797 (2017)