# Filter Style Transfer between Photos

Jonghwa Yim, Jisung Yoo⋆, Won-joon Do, Beomsu Kim, and Jihwan Choe

Visual Solution Lab., Samsung Electronics, South Korea
{jonghwa.yim, jisung.yoo, wonjoon.do, bs8207.kim,
jihwan.choe}@samsung.com
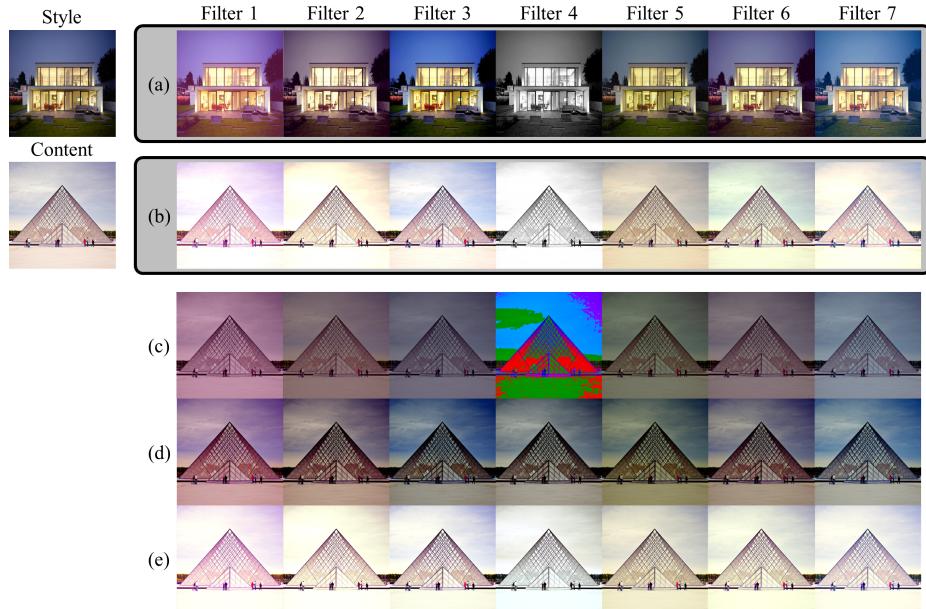
**Fig. 1. Filter style transfer results.** Given reference images with arbitrary filters applied (a), the filter styles can be transferred to a new image with our model. While (b) is the ground-truth, (c)-(e) show the results of color transfer [24], photorealistic style transfer: WCT2 [32], and ours, respectively.

**Abstract.** Over the past few years, image-to-image style transfer has risen to the frontiers of neural image processing. While conventional methods were successful in various tasks such as color and texture transfer between images, none could effectively work with the custom filter effects that are applied by users through various platforms like Instagram. In this paper, we introduce a new concept of style transfer, Filter Style Transfer (FST). Unlike conventional style transfer, new technique

---

⋆ Corresponding Author

FST can extract and transfer custom filter style from a filtered style image to a content image. FST first infers the original image from a filtered reference via image-to-image translation. Then it estimates filter parameters from the difference between them. To resolve the ill-posed nature of reconstructing the original image from the reference, we represent each pixel color of an image to class mean and deviation. Besides, to handle the intra-class color variation, we propose an uncertainty based weighted least square method for restoring an original image. To the best of our knowledge, FST is the first style transfer method that can transfer custom filter effects between FHD image under 2ms on a mobile device without any textual context loss.

## 1   Introduction

Stylizing an image with characteristics of other stylized images has long been a difficult problem in Computer Vision. Beyond simple editings, people's desire to grand artistic feelings to their pictures has increased. For this reason, a tool that can stylize their photos in a unique way is highly desired.

There have been several studies addressing technical solutions for image-to-image style or content transfer. Reinhard et al. [24] is one of the pioneering attempts where mean and variance of RGB color distribution from a source image were used to apply the color scheme to a target image, but with limited success obtaining enough similarity between two images. Others [23, 26–28] tried to improve results by using various mathematical approaches to treat color distribution but failed to consider the semantics of pictures during the process. Moreover, their methods transferred objects' inherent colors as well, limiting their methods in the assumption that scene components of the two images must be similar.

More recently, taking advantage of the advent of Deep Neural Network, more sophisticated applications of image-to-image style transfer became possible. Style transfer [7] encoded not only color but also shapes and textures. After that, many following works branched out to further improve the accuracy and efficiency of the style transfer. Some researches [12, 20, 35] were related to domain transfer, which transfers styles between different image domains such as semantic-labels to street-scene, aerial to map, and sketch to photo. However, since most domain transfer approaches aimed to move input images' distributions close to the target domain, the output of them does not explicitly reflect the style of a single image. Some other researches [18, 21, 25, 32] introduced methods to transfer photorealistic styles from a single style image, but they required a large dataset for training leading to a high computational cost. Even with the high processing time, they often displayed undesirable transfers of colors and textures due to fundamentally implicit actions of deep neural networks. In this case, it is difficult to identify causes and solutions, which is a significant hurdle when commercializing the approaches.

Meanwhile, some researches suggested automated photograph editing to enhance the overall quality [1–5, 8, 11, 14, 15, 22, 29, 30] or control exposure [10, 31, 33, 34]. All of these researches showed considerable progress on automated image editing, but they also required large datasets to train an image enhancement model. More importantly, they only followed a predefined editing rule like High Dynamic Range (HDR). Some focused on the extraction of photo-editing-parameters directly [1, 2, 8, 22], while [5] focused on learning image enhancement using GAN [9] to generate HDR output. A method in [2] suggested parameter extraction from a neural network, but it was not a single-stage and showed limited performance. Also, efforts to model polynomial functions in the previous studies [1, 2] may suffer from high-order variables' fitting issues as well as they still limited their methods to predefined editing rules. There have been a few efforts to adopt reinforcement learning [10, 33] to train enhancement policies. Despite all the efforts, all the aforementioned methods were not able to extract filter parameters from an already stylized image and require the original version of the stylized image to enhance the target image. Such limitations prevented previous studies from fully satisfying commercial needs.

With the increased accessibility of mobile phones and the internet, these days, people spend even more time on social media. As a result, many photo-editing applications have been developed and are widely used with various stylizing filters to give special effects on photos taken. To the best of our knowledge, however, there has not been an attempt to extract custom filter effects from a stylized photo. In this study, a mathematical formulation of custom filter extraction and its application to new photos are presented. FST is quite efficient without requiring expensive computing time, even in a low-end mobile device; so the application can be easily adopted and used in our fast-moving social networking environment.

## 2   Method Overview

Fig. 2 shows an overview of the proposed method in this research. It comprises extracting custom photo filters from a single reference image ($I$) and applying them to a new one ($X$). Our method restores the original image ($\hat{I}$) from $I$, which is called *defilterization* in this paper. Then, using two images, a filter parameter $w$ is obtained, which is called *filter style estimation*. Lastly, using $w$, a designed filter function $f_w$ can be used to filter the user's original image ($X$) to newly-stylized image ($Y$).

## 3   Defilterization

In Fig. 2, the stylized reference input $I$ is a projected image from the original image $\acute{I}$ using the filter-applying function $f_w$, leading to

$$I = f_w(\acute{I}). \tag{1}$$
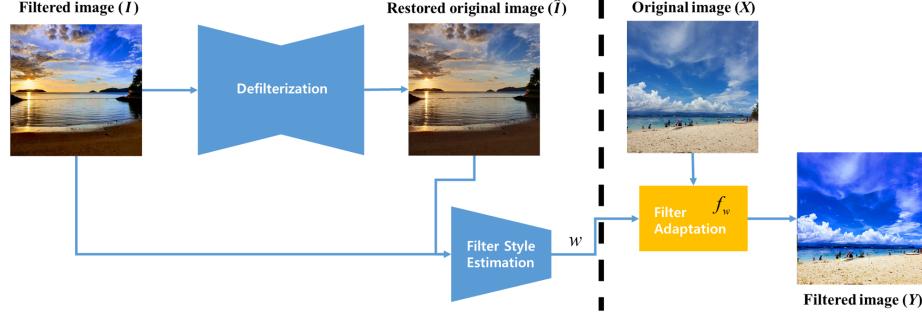
**Fig. 2. System overview to extract the photo editing parameter and apply it to the new input images.** $I$ is a filtered or edited image that we want to extract filter style and $w$ represents the parameters of the filter style of $I$. The system is initially black-box and must be designed appropriately to infer standalone parameters.

To determine $f_w$, the relationship between the pair of the original image $\acute{I}$ and stylized image $I$ needs to be investigated. From (1), we know that

$$\acute{I} = f_w^{-1}(I) \tag{2}$$

Assume that there is a collection $S$ of $M$ stylized images, $S = \{I_1, I_2, \ldots, I_M\}$. Each image $I$ consists of $K$ object segments, like the sky, cow, grass, etc., and each object segment can be represented its vectorized form $\boldsymbol{o}$, such that

$$I = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \boldsymbol{o}_3, \ldots, \boldsymbol{o}_K\} \tag{3}$$

where $\boldsymbol{o}_k$ is a vector of colors of the flattened pixels in the $k$-th object segment. Then the original image of $I$ can also be represented as a set of the original object segments, such that

$$\acute{I} = \{\acute{\boldsymbol{o}}_1, \acute{\boldsymbol{o}}_2, \acute{\boldsymbol{o}}_3, \ldots, \acute{\boldsymbol{o}}_K\} \tag{4}$$

where $\acute{\boldsymbol{o}}_k$ is the original colors before stylized. If an implicit object, $\acute{\boldsymbol{o}}_k$, has some class label, $cls$, and its mean color can be obtained by averaging all pixel colors in $cls$ throughout the dataset $S$, then, $\acute{\boldsymbol{o}}_k$ can be expressed with the mean color value of the class, $\tilde{\boldsymbol{o}}_{k,cls}$, and pixel-wise color deviations, $\Delta_{k,cls}$. Note that we start to explain from the implicit object level to introduce the class label and its mean color. Also, there is a numerical error term $\epsilon_k$ due to imperfect restoration of original object colors. Therefore $\acute{\boldsymbol{o}}_k$ can be expressed as

$$\acute{\boldsymbol{o}}_k = \tilde{\boldsymbol{o}}_{k,cls} + \Delta_{k,cls} = f_w^{-1}(\boldsymbol{o}_k) + \boldsymbol{\epsilon}_k. \tag{5}$$

The distance between the restored image, $f_w^{-1}(I)$, and the true original, $\acute{I}$, can be described as the sum of the squared distance between restored and true objects.

$$Distance(f_w^{-1}(I), \acute{I}) = \sum_{k=1}^{K} \|f_w^{-1}(\boldsymbol{o}_k) - \acute{\boldsymbol{o}}_k\|^2 \tag{6}$$

Then our problem to find the original image becomes a minimization problem. At a pixel level, the objective function of the minimization process becomes

$$\arg\min_{f_w^{-1}} \frac{1}{N} \sum_{k=1}^{K} \left[ \sum_{c \in \boldsymbol{o}_k} \left( f_w^{-1}(c) - \acute{c} \right)^2 \right],$$ (7)

where $N$ is the number of pixels in $I$. The sum of the differences of $\boldsymbol{o}_k$ can be reformulated by merging two summations in (7). Then, the distance is a sum of the squared difference between $f_w^{-1}(c)$ and $\acute{c}$. Converting (5) into pixel-level representation and substituting $\acute{c}$ into its mean and deviations lead to

$$\arg\min_{f_w^{-1}} \frac{1}{N} \sum_{c \in I} \left( f_w^{-1}(c) - \tilde{c}_{cls} - \varDelta_{c,cls} \right)^2$$ (8)

where $\tilde{c}_{cls}$ is an element of $\tilde{\boldsymbol{o}}_{k,cls}$. Since the objective function (7) and (8) corresponds to the error criterion of the neural network, especially autoencoder, where the pixel differences can be calculated after forward-passing the input image, we now let $f_w^{-1}$ be an autoencoder network and train to infer $\acute{c}$ over the dataset $S$. Then we can expect that the trained autoencoder can restore the original image considering implicit semantic, $cls$. Thus, the objective function over the entire dataset $S$ of $M$ equal-sized images is

$$\arg\min_{f_w^{-1}} \sum_{m=1}^{M} \left[ \sum_{c \in I_m} \left( f_w^{-1}(c) - \tilde{c}_{cls} - \varDelta_{c,cls} \right)^2 \right].$$ (9)

By definition, $\sum_{c \in S_{cls}} \varDelta_{c,cls} = 0$ where $S_{cls}$ is a subset of $S$ that belongs to a class label $cls$. With this definition, after some calculation, eq. (9) becomes

$$\arg\min_{f_w^{-1}} \sum_{m=1}^{M} \left[ \sum_{c \in I_m} \left( \left( f_w^{-1}(c) - \tilde{c}_{cls} \right)^2 + 2\epsilon_c \varDelta_{c,cls} - \varDelta_{c,cls}^2 \right) \right].$$ (10)

Since $\varDelta^2$ is a constant for a given dataset $S$, the minimization becomes

$$\arg\min_{f_w^{-1}} \sum_{m=1}^{M} \left[ \sum_{c \in I_m} \left( \left( f_w^{-1}(c) - \tilde{c}_{cls} \right)^2 + 2\epsilon_c \varDelta_{c,cls} \right) \right].$$ (11)

Then $f_w^{-1}$ learns to restore image toward the mean, between the mean and original. Therefore, one can geometrically assume that $\epsilon_c$ at the optimum point is smaller than and proportional to $\varDelta_{c,cls}$. Since deducing $\varDelta_{cls}$ solely from a single image is an ill-posed problem, our method minimizes the influence of the inevitable error $\epsilon_c$ by collecting pixels during regression in chapter 4.1.

## 4  Filter Style Estimation

### 4.1  Filter Parameterization

Most of the image filtering and editing can be built with three operations – brightness, contrast, and color controls. Even though there are some local operations such as Vignetting, for simplicity, we have not considered those in this

study. In general the three primary operations can be expressed as linear or polynomial functions for input image $x$ and output image $y$;

$$\text{Brightness} \qquad y_1 = x + c \qquad\qquad (12)$$

$$\text{Contrast} \qquad y_2 = ax + b \qquad\qquad (13)$$

$$\text{Color} \qquad y_3 = \sum_{i=1}^{\alpha} \left( e_i x_i^3 + f_i x_i^2 + g_i x_i \right) \qquad (14)$$

where $\alpha$ is the number of color channels, three (i.e. RGB) in our case. After adding up $y_1$, $y_2$, and $y_3$ and expressing parameters as $\boldsymbol{\beta}$, the three operations becomes

$$y_\gamma = \beta_{\gamma,0} + \sum_{i=1}^{3} \left( \beta_{\gamma,i1} x_i + \beta_{\gamma,i2} x_i^2 + \beta_{\gamma,i3} x_i^3 \right). \qquad (15)$$

Note that we repeatedly calculate $y_\gamma$ over the output color channel, i.e. $\gamma \in \{R, G, B\}$. Hereafter, we omit $\gamma$ for brevity. Since (15) represents global editing operations, using this, we model the parametric function $f_{\boldsymbol{\beta}}^*$ of $f_w$ in the regarding $\boldsymbol{\beta}$. Thus, with the original and the reference image, we can approach filter parameter extraction, obtaining $\boldsymbol{\beta}$ in (15), as a nonlinear regression problem operated at every pixel of an image. Hence the minimization target $E(\boldsymbol{\beta})$ is

$$E(\boldsymbol{\beta}) = \sum_{n=1}^{N} \left( y_n - f_{\boldsymbol{\beta}}^*(\acute{x}_n) \right)^2 \qquad (16)$$

where $(\acute{x}, y)$ is a color pair in $(\acute{I}, I)$. After applying (5), (16) becomes (17). In a normalized color domain, recalling the geometrical interpretation of (11), $\epsilon$ becomes small, and thus the high order of $\epsilon$ becomes negligibly small. After some calculation and with the assumption that $\epsilon$ is proportional to $\Delta$, (17) becomes

$$E(\boldsymbol{\beta}) = \sum_{n=1}^{N} \left( y_n - f_{\boldsymbol{\beta}}^*(f_w^{-1}(y_n) + \epsilon_n) \right)^2 \qquad (17)$$

$$\approx \sum_{n=1}^{N} \left( y_n - f_{\boldsymbol{\beta}}^*(f_w^{-1}(y_n)) \right)^2. \qquad (18)$$

Since high order terms of $\epsilon_n$ are ignored, (18) is a rough approximation on a single image. Due to the ill-posed nature of the problem, instead, we propose uncertainty-based regression in chapter 4.2 to alleviate the error of rough approximation. To this end, we set the problem to weighted least squares and solved it using quasi-Newton optimization. Note that eq. (18) works well when $\sum_{c \in I} \Delta_{c,cls}$ is close to 0.

Additionally, our method can also provide results similar to color transfer, depending on filter parameters. If there are insufficient samples in the RGB color domain in a stylized image, it would not be easy to infer the coefficients to cover

the absence of samples. In this case, if the Channel Correlation term (CC) is added, those colors can be transferred to other colors correlated with RG, RB, GB values as below.

$$y_\gamma = \beta_{\gamma,0} + \sum_{i=1}^{3} \left( \beta_{\gamma,i1}x_i + \beta_{\gamma,i2}x_i^2 + \beta_{\gamma,i3}x_i^3 \right) + \beta_{\gamma,1}x_1x_2 + \beta_{\gamma,2}x_1x_3 + \beta_{\gamma,3}x_2x_3 \tag{19}$$

Then the result becomes more like color transfer than FST. For example, in Fig. 3, green color is absent in stylized image. Therefore, green is transferred to another color in the result. More details will be presented in chapter 5.4.
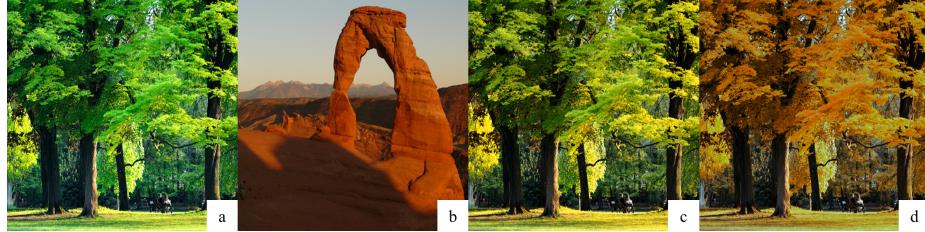


**Fig. 3. The result comparison with and without correlation term.** (a) is an input image. (b) is a stylized reference image. (c) and (d) are FST results using (15) and (19), respectively.

### 4.2   Uncertainty-based Adaptive Filter Regression

After training $f_w^{-1}$, for any single filtered style image, we depend on a trained neural network to get the restored image and to regress the approximate function $f_{\boldsymbol{\beta}}^*$. In the process, there are high order terms of $\epsilon$ that causes the filter estimation error in the previous chapter. Due to the lack of evidence to directly minimize the error, we propose a roundabout method using the uncertainty of the inference and lower the weight where $\epsilon$ is expected high.

The error term is inherently non-negligible in our case since the function inferred $\Delta_{c,cls}$ solely from a single image. To be more specific, in the single inference of $I$, from each pixel $c$, the uncertainty of $f_w^{-1}(c)$ would be high when the implicit class variance $Var(\Delta_{c,cls})$ is high over the entire set $S$ (aleatoric uncertainty). That means if the variance of the deviation of $cls$ is high over $S$, the function $f_w^{-1}(c)$ is likely to give larger $\epsilon$. Moreover, $f_w^{-1}(c)$ would be more uncertain as $\Delta_{c,cls}$ is increased (epistemic uncertainty). In this case, the weight of unsure pixels should be lower when regressing the approximate function, $f_{\boldsymbol{\beta}}^*$.

The error term is independent over pixels. Therefore, we compute variance term same as combined uncertainty, a combination of epistemic uncertainty from Mean Standard Deviation (Mean STD) in the earlier study [6] and aleatoric

uncertainty in [16] in recovering the original image. Then for every pixel, the inverse of uncertainty, written as $\boldsymbol{\Omega}^{-1}$, is used as a weight of the least squares criterion. Then the general form of the solution to our regression problem is

$$\boldsymbol{\beta} = (\boldsymbol{X}^T \boldsymbol{\Omega}^{-1} \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{\Omega}^{-1} \boldsymbol{y} \tag{20}$$

In practice, our uncertainty-based regression can be achieved by multiplying $\boldsymbol{\Omega}^{-1/2}$ to both of the $\boldsymbol{X}$ and $\boldsymbol{y}$ followed by quasi-Newton optimization. Note that $\boldsymbol{X}$ and $\boldsymbol{y}$ are the design matrix that consists of stacked polynomial vectors of the restored original colors and the vector of the filtered colors, respectively.

### 4.3   Regularization

With the methods described in previous chapters, we are now able to estimate the parameters of filter style from single image input. However, there are two problems with this unrefined algorithm. Firstly, when we do not have enough plots around each extremum of color space, the regression function is left to vary dramatically outside of plots. Like the blue line shown in Fig. 4, a polynomial function curves very fast without the basis of plots, leading the extrema transformed to unfavorable values. So, for the new user input image $X$, stylized image $Y$ often show clipping or extreme colors around the extremes. The second problem is that the output can sometimes be visually unnatural as the regression function severely deviates from linear, leaving the regression process vulnerable to specific colors, which are exceptionally scarce but saturated in the pairs of the stylized and inferred original.

To relieve the above symptoms, we design and add regularization term in the regression function. To deal with the first phenomenon, we regularize the function to be close to $(0,0)$, $(1,1)$, respectively, when the color range is normalized. Hence, an L2 penalty is added and penalizes the function when it starts to diverge from 1 and 0. For each output channel $\gamma$,

$$R_{1,\gamma} = \beta_{\gamma,0}^2 + \left\{ \left[ \beta_{\gamma,0} + \sum_{i=1}^{3} (\beta_{\gamma,i1} + \beta_{\gamma,i2} + \beta_{\gamma,i3}) \right] - 1 \right\}^2 \tag{21}$$

After adding (21), the nonlinear function looks like a red line instead of a blue line in Fig. 4.

For the second symptom, we add an L2 penalty on the coefficients of high order terms. Empirically we found out that imposing the L2 penalty only on different sources of colors is visually good, rather than imposing L2 on all color sources.

$$R_{2,\gamma} = \sum_{i \neq \gamma} \left( \beta_{\gamma,i2}^2 + \beta_{\gamma,i3}^2 \right) \tag{22}$$
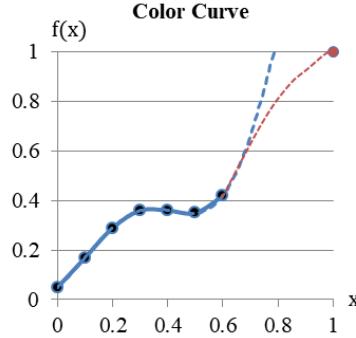
**Fig. 4. An example of a nonlinear function that has unfavorable extrema matching in a color transfer curve.** Since there is no color sample near $x = 1$, the function can diverge as shown by the blue dotted line. Instead, we can add a regularization term to guide nonlinear function into a red line.

After adding two regularization terms (21) and (22), the error of regression function becomes

$$\acute{E}(\boldsymbol{\beta}_\gamma) = E(\boldsymbol{\beta}_\gamma) + \lambda R(\boldsymbol{\beta}_\gamma), \text{ where} \tag{23}$$

$$R(\boldsymbol{\beta}_\gamma) = \beta_{\gamma,0}^2 + \left\{ \left[ \beta_{\gamma,0} + \sum_{i=1}^{3} (\beta_{\gamma,i1} + \beta_{\gamma,i2} + \beta_{\gamma,i3}) \right] - 1 \right\}^2 + \sum_{i \neq \gamma} \left( \beta_{\gamma,i2}^2 + \beta_{\gamma,i3}^2 \right) \tag{24}$$

In Fig. 5, we show the difference in result images by introducing our regularization term. In this figure, the result when regularization weight is 0 shows clipping around the ground region, while the full use of the regularization does not exhibit this behavior. Note that $\lambda$ can be obtained by grid-search.
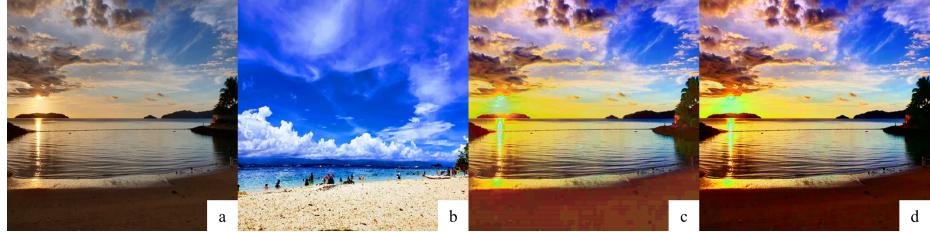


**Fig. 5. An example of the regularization effect.** Regularization term yields a stable result where there is less or no clipping. (a) is an input image, (b) is a filtered style image, and (c) and (d) are results of FST without and with regularization terms, respectively.

## 5  Experiment

### 5.1  Dataset

To generalize the defilterization network $f_w^{-1}$ for various scenes, we require a large-scale dataset with lots of classes. One of the most popular image datasets, MSCOCO [19], is widely used and contains more than 110K images with 80 object categories. However, this dataset does not contain filtered images. Therefore, to generate filtered images, we applied various types of real and arbitrary synthetic photo filters to the dataset. Initially, a filtered dataset was generated by posing 26 real Instagram filters using publicly available source code in CSSgram [17]. In addition, synthetic filtered images were added by posing random color, contrast, and brightness six times. Based on this dataset, we trained the defiliterization network and tested our proposed method.

To check the dataset dependecy, we also prepared 99 private photos and 17 unseen real filters from one of the camera application in Android Play Store. As shown in the experiment in the next chapter, the proposed method, FST, can successfully transfer filter effects from a single image to the new input, even the filters unseen in the training phase.

### 5.2  Evaluation

Firstly we prepared the architecture of image-to-image translation, introduced in [13], as a defilterization function $f_w^{-1}$. Then, we trained $f_w^{-1}$ using the combined dataset and fully synthetic dataset (self-supervised learning) until the test MSE is saturated. Note that the filter transferred output would show better results with a better choice of defilterization network and more synthetic dataset generation, but we leave it as further work.

To validate our proposed method, we randomly chose 100 images from the MSCOCO validation set, and selected 18 real filters out of 26 filters to generate 1800 filtered images. We excluded eight filters that have a noticeable vignetting effect, which is outside of the scope of the current study. Then, FST was performed on the remaining validation images, and the result was compared with the ground truth images, which were directly generated by applying filters. Quantitative and qualitative results are given in Table 1 and Fig. 6, respectively. To further test our method on unseen dataset, we also evaluated the proposed method on private photos with 43 filters (17 unseen filters and 26 Instagram filters). The result is given in Table 2.

### 5.3  Comparison with Style Transfer

Our problem definition is inherently different from conventional style transfer researches. Style transfer seeks the transfer of texture, color, and even abstract concepts, while our method targets the transfer of photo editing or filter effects applied to an image. Although photorealistic stylization approaches [18,32] show

**Table 1. Quantitative evaluation.** The MSCOCO validation set with 18 Instagram filters is used for evaluation. Our method supports a few variations. Our method can have (R): regularization, (AU): Aleatoric uncertainty, (U): combined uncertainty, (CC): color correlation. Note that for WCT2, we gave option that uses features from decoder and skip-connection since it performs the best. Note that lower $\Delta E_{00}^*$ (a.k.a., Delta-E 2000) is better.

| Methods | PSNR | $\Delta E_{00}^*$ |
|---|---|---|
| **Ours** | **25.226** | **6.660** |
| **Ours (w. U, CC)** | **24.931** | **6.725** |
| **Ours (w. AU)** | **25.438** | **6.427** |
| **Ours (w. U)** | **25.495** | **6.394** |
| **Ours (w. R, U)** | **26.093** | **6.148** |
| WCT2 [32] | 16.473 | 17.516 |
| Color Transfer [24] | 7.325 | 34.914 |

**Table 2. Quantitative evaluation on private photos with 43 filters.**

| Methods | Ours (w. R, AU) | Ours (w. R, U) | WCT2 [32] | Color Transfer [24] |
|---|---|---|---|---|
| PSNR | **25.814** | **25.850** | 17.234 | 6.985 |
| $\Delta E_{00}^*$ | **5.881** | **5.854** | 15.723 | 35.123 |



| Style Original | Filtered Style | Input | Output | Ground Truth |
|---|---|---|---|---|
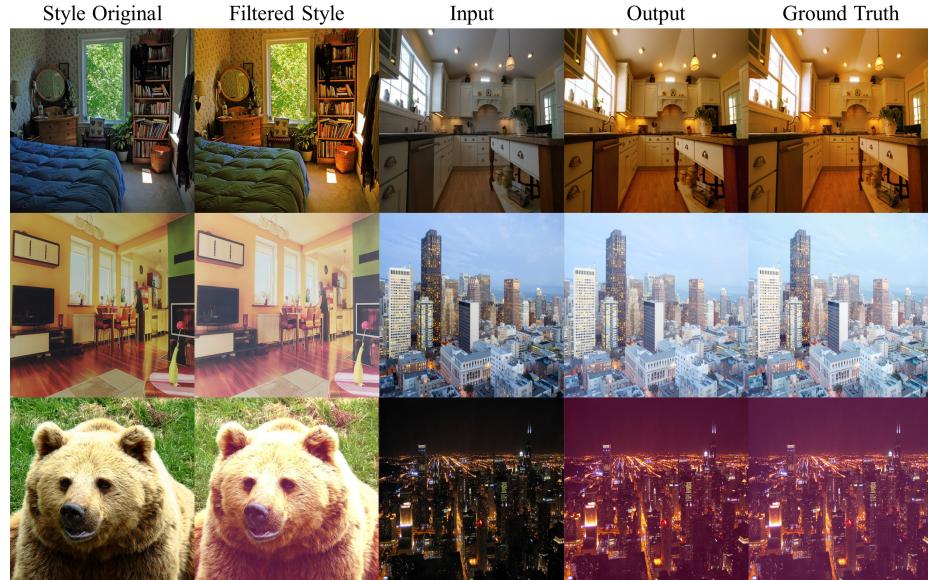
**Fig. 6. Qualitative results of FST.** We also included the originals of filtered style images, which were not given during inference.

attractive results in terms of structure preservation, they tend to directly transfer color distribution from the style to content images, not transferring filter information of the style image. Nonetheless, we compared our proposed FST with photorealistic style transfer, since style transfer is the most similar task.

To validate the effectiveness of the proposed method, we compare it with two types of photorealistic style transfer methods based on conventional linear color distribution transfer [24] and structure-preserved style transfer based on a high-frequency component skip, which is the state-of-the-art in photorealistic style transfer [32]. As shown in Fig. 7, both approaches tend to directly reflect the color distribution of the style image to the content image while FST transfers filter style only. In case of color transfer, if style image consists of achromatic colors mostly, it often generates visually unpleasing and questionable results. Furthermore, in terms of computational complexity, while conventional style transfer techniques take several hundred milliseconds, our solution can transfer filter style to the content image in less than a few milliseconds. Moreover, FST can transfer stylish effects as well as unseen filters. Detailed results are shown in Fig. 8.



(a). Inputs        (b). Color Transfer        (c). WCT2        (d). Ours (w/o. CC)        (e). Ours (w. CC)
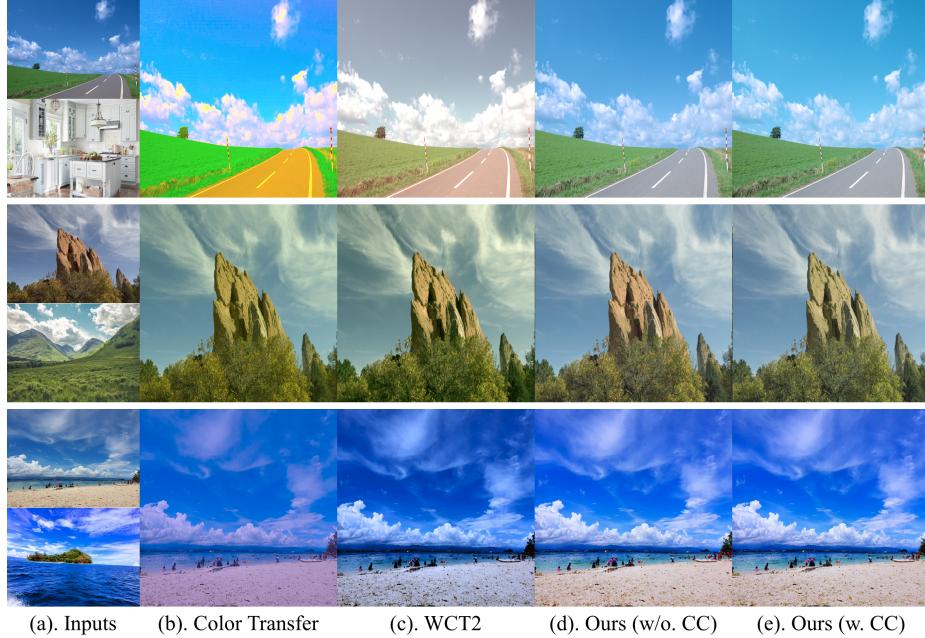
**Fig. 7. The results of FST with channel correlation.** Given (a) an input pair (top: content, bottom: style), the results of (b) color transfer [24], (c) WCT2 [32], (d) and (e) ours (FST) are shown. With CC (color correlation) term, the proposed method can also transfer colors of contents, whitish color in the first row, green forests in the second row, and bluish color in the third row, for example.

**Fig. 8. More results from the proposed FST using stylish photos.**

### 5.4   Application

**A trade-off between Filter and Color Style.**    In chapter 4.1, the approximate filter-applying function is modeled as a polynomial form (15) to transfer filter styles. However, in addition to filter style, more dramatic effects may be required upon requests. In this case, by adding a correlation term to filter-applying function as shown in (19), FST can transfer some colors that are not present in the reference style image to neighbor colors at the expense of quantitative accuracy. As shown in Fig 3, results look more like color style transfer than the original FST.

**Real-time Filter Transfer on Mobile Device.**    Along with the satisfactory results of our method, it is designed to run real-time on a mobile environment, where there is a severe restriction on computational power. FST takes most of its time on filter parameter extraction, and it costs as much as the inference time of autoencoder, plus nonlinear regression. However, once the parameters are obtained, transferring the filter style onto the new input can be done almost instantly. The processing times are firstly measured on a PC, and compared with WCT2 and Color Transfer. Results are given in Table 3. Furthermore, in a mobile environment, our approach performs with 900ms on average on Qualcomm Snapdragon 855 to process an FHD image. The processing time on a smartphone is given in Table 4. Note that once the filter is extracted, RGB Look Up Table (LUT) can be precomputed and stored on the device to shorten the processing time. Then, in the run-time, pixels are matched to new values using LUT to generate a stylized output image, which requires less than 2ms to transfer FHD images.

## 6   Conclusion and Future Work

To the best of our knowledge, this is the first study on the real-time filter transfer between two real photos. Our approach resolves a new task called FST (Filter

**Table 3. Run-time comparison on a PC.** Steps are divided into two; filter extraction and application. Tests were done using the machine with Python Numpy, Nvidia GTX 1080ti, and Intel i7-8700 CPU. We used 256x256 image for filter extraction and 1920x1080 (FHD) for filter application. Note that excluding Epistemic Uncertainty (EU) in the proposed method can shorten the time required for filter extraction. We used 10 MC dropouts for EU.

| Methods | Filter extraction | Filter application |
|---|---|---|
| **Ours (w/o. EU)** | **28ms** | **86ms** |
| **Ours (full ver.)** | **132ms** | **86ms** |
| WCT2 [32] | 16ms | 943ms |
| Color Transfer [24] | 49ms (single stage) | |

**Table 4. Run-time on a smartphone.** We also measured our method in a mobile environment, Samsung Galaxy S10. We used 256x256 image for filter extraction and 1920x1080 (FHD) for filter application. The filter application is much faster than the PC version since the mobile version uses GPU parallel processing, while the PC version partially uses Numpy CPU. Note that we excluded EU from the mobile version.

| Methods | Filter extraction | Filter application |
|---|---|---|
| **Ours, mobile version** | **900ms** | **2ms** |

Style Transfer), transferring custom filter operation between images, which is different from previous works of style transfer. Although style transfer methods yield reasonable outputs in some cases, it does not consistently generate pleasant outputs in every case and may require an additional effort on tuning the result. Moreover, arbitrary photorealistic style transfer still provides degraded or ruined texture, which is not desirable in photo filter extraction.

In a mobile environment, once filter parameters acquired, FST consistently runs within 2ms to transfer FHD previews in camera applications, which shows exceptional real-time performance. Moreover, our solution can also perform similar to color transfer depending on the regression function, but it still creates more natural output than any other existing color transfer method.

In the proposed method, a defilterization network can be replaced by any other network structures. However, note that the performance of the network directly relates to the performance of filter transfer. In the future, we therefore plan to work on improving the performance of the defilterization network to extract the originals from filtered images more accurately. For implicit learning of semantical objects, it may require to train class labels explicitly as well as decoder part. If the defilterization network can perfectly extract the original image, the final result of the filter transfer would be more reliable.

# References

1. Bianco, S., Cusano, C., Piccoli, F., Schettini, R.: Content-preserving tone adjustment for image enhancement. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
2. Bianco, S., Cusano, C., Piccoli, F., Schettini, R.: Learning parametric functions for color image enhancement. In: International Workshop on Computational Color Imaging. pp. 209–220. Springer (2019)
3. Bychkovsky, V., Paris, S., Chan, E., Durand, F.: Learning photographic global tonal adjustment with a database of input/output image pairs. In: CVPR 2011. pp. 97–104. IEEE (2011)
4. Chandakkar, P.S., Li, B.: Joint regression and ranking for image enhancement. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 235–243. IEEE (2017)
5. Chen, Y.S., Wang, Y.C., Kao, M.H., Chuang, Y.Y.: Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6306–6314 (2018)
6. Gal, Y., Islam, R., Ghahramani, Z.: Deep bayesian active learning with image data. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1183–1192. JMLR. org (2017)
7. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2414–2423 (2016)
8. Gharbi, M., Chen, J., Barron, J.T., Hasinoff, S.W., Durand, F.: Deep bilateral learning for real-time image enhancement. ACM Transactions on Graphics (TOG) **36**(4), 1–12 (2017)
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
10. Hu, Y., He, H., Xu, C., Wang, B., Lin, S.: Exposure: A white-box photo post-processing framework. ACM Transactions on Graphics (TOG) **37**(2), 1–17 (2018)
11. Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., Van Gool, L.: Dslr-quality photos on mobile devices with deep convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3277–3285 (2017)
12. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017)
13. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision. pp. 694–711. Springer (2016)
14. Kang, S.B., Kapoor, A., Lischinski, D.: Personalization of image enhancement. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 1799–1806. IEEE (2010)
15. Kaufman, L., Lischinski, D., Werman, M.: Content-aware automatic photo enhancement. In: Computer Graphics Forum. vol. 31(8), pp. 2528–2540. Wiley Online Library (2012)
16. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? In: Advances in neural information processing systems. pp. 5574–5584 (2017)

17. Kravets, U.: Cssgram (2016), https://github.com/una/CSSgram

18. Li, Y., Liu, M.Y., Li, X., Yang, M.H., Kautz, J.: A closed-form solution to photorealistic image stylization. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 453–468 (2018)

19. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)

20. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: Advances in neural information processing systems. pp. 700–708 (2017)

21. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4990–4998 (2017)

22. Omiya, M., Simo-Serra, E., Iizuka, S., Ishikawa, H.: Learning photo enhancement by black-box model optimization data generation. In: SIGGRAPH Asia 2018 Technical Briefs. p. 7. ACM (2018)

23. Pitié, F., Kokaram, A.C., Dahyot, R.: Automated colour grading using colour distribution transfer. Computer Vision and Image Understanding **107**(1-2), 123–137 (2007)

24. Reinhard, E., Adhikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Computer graphics and applications **21**(5), 34–41 (2001)

25. Sheng, L., Lin, Z., Shao, J., Wang, X.: Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8242–8250 (2018)

26. Tai, Y.W., Jia, J., Tang, C.K.: Local color transfer via probabilistic segmentation by expectation-maximization. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1, pp. 747–754. IEEE (2005)

27. Welsh, T., Ashikhmin, M., Mueller, K.: Transferring color to greyscale images. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques. pp. 277–280 (2002)

28. Xiao, X., Ma, L.: Color transfer in correlated color space. In: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications. pp. 305–309 (2006)

29. Yan, J., Lin, S., Bing Kang, S., Tang, X.: A learning-to-rank approach for image color enhancement. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2987–2994 (2014)

30. Yan, Z., Zhang, H., Wang, B., Paris, S., Yu, Y.: Automatic photo adjustment using deep neural networks. ACM Transactions on Graphics (TOG) **35**(2), 1–15 (2016)

31. Yang, H., Wang, B., Vesdapunt, N., Guo, M., Kang, S.B.: Personalized exposure control using adaptive metering and reinforcement learning. IEEE transactions on visualization and computer graphics **25**(10), 2953–2968 (2018)

32. Yoo, J., Uh, Y., Chun, S., Kang, B., Ha, J.W.: Photorealistic style transfer via wavelet transforms. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9036–9045 (2019)

33. Yu, R., Liu, W., Zhang, Y., Qu, Z., Zhao, D., Zhang, B.: Deepexposure: Learning to expose photos with asynchronously reinforced adversarial learning. In: Advances in Neural Information Processing Systems. pp. 2149–2159 (2018)

34. Yuan, L., Sun, J.: Automatic exposure correction of consumer photographs. In: European Conference on Computer Vision. pp. 771–785. Springer (2012)

35. Zhu, J.Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O., Shechtman, E.: Toward multimodal image-to-image translation. In: Advances in neural information processing systems. pp. 465–476 (2017)