IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 14, 2019, accepted May 31, 2019, date of publication June 17, 2019, date of current version July 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923400

Gamut-Extension Methods Considering Color Information Restoration

MASARU TAKEUCHI[1], (Member, IEEE), YUSUKE SAKAMOTO[1], RYOTA YOKOYAMA[1], HEMING SUN[2], (Member, IEEE), YASUTAKA MATSUO[3], (Member, IEEE), AND JIRO KATTO[1], (Member, IEEE)

[1]Department of Computer Science and Communication Engineering, Waseda University, Tokyo 1698555, Japan
[2]Waseda Research Institute for Science and Engineering, Tokyo 1698555, Japan
[3]Science & Technology Research Laboratories, Japan Broadcasting Corporation (NHK), Tokyo 1578510, Japan

Corresponding author: Masaru Takeuchi (takeuchi@katto.comm.waseda.ac.jp)

ABSTRACT Recently, ultra high definition television (UHDTV) services have become popular using satellites and the Internet. However, there are expansive volumes of high definition television (HDTV) and standard definition television (SDTV) contents held by broadcasting companies and in storage devices. Herein we propose two color space conversion (also known as gamut mapping) methods from BT.709 (used for current HDTV broadcast) to BT.2020 (used for UHDTV broadcast) that restore or estimate lost color information. One of our methods anisotropically diffuses the BT.709 chromaticities with regard to the direction of the original chromaticities in the BT.2020 color space, generating chromaticities out of BT.709 gamut. The other learns an end-to-end conversion method from a BT.709 image to a BT.2020 image and restores lost color information using convolutional neural network (CNN). Using these methods along with BT.709 images, we obtain BT.2020 images with chromaticities from the BT.709 color gamut.

INDEX TERMS Image filtering, image color analysis, image reconstruction, convolutional neural network.

I. INTRODUCTION

Recently, research and development toward practical Ultra High Definition Television (UHDTV) next-generation broadcasts have been advanced [1]. As shown in Fig. 1, the definitions of the video signals of High Definition Television (HDTV) and UHDTV differ. Hence, their richness in spatial, temporal, and amplitudinal characteristics, dynamic range, and color gamut varies. Among these characteristics, the color gamut enables video signals to represent nearly true colors with BT.2020 [2]. This color space covers almost all chromaticities in the real world. On the other hand, existing contents recorded with the BT.709 [3] color space do not possess chromaticities outside the BT.709 gamut. Moreover, such contents cannot be converted into their original chromaticities with a simple linear color space conversion.

While there are Super-resolution methods to generate high resolution images from low resolution ones and Motion- interpolation methods to generate high-frequency video sequences from low-frequency ones, there are also gamut-extension methods generate wide color gamut images from narrow ones. Previously we proposed a super resolution method using a wavelet transform and an affine transform [4] as well as an SDR to HDR conversion method using Convolutional Neural Network (CNN) [5].

Herein we consider two methods of color space conversion from BT.709 to BT.2020 using anisotropic chromaticity diffusion and CNN.

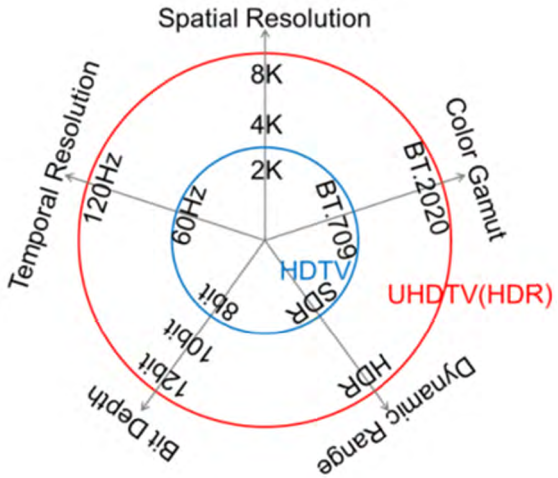


FIGURE 1. Evolution of hardware specifications and video format.

The associate editor coordinating the review of this manuscript and approving it for publication was Shagufta Henna.

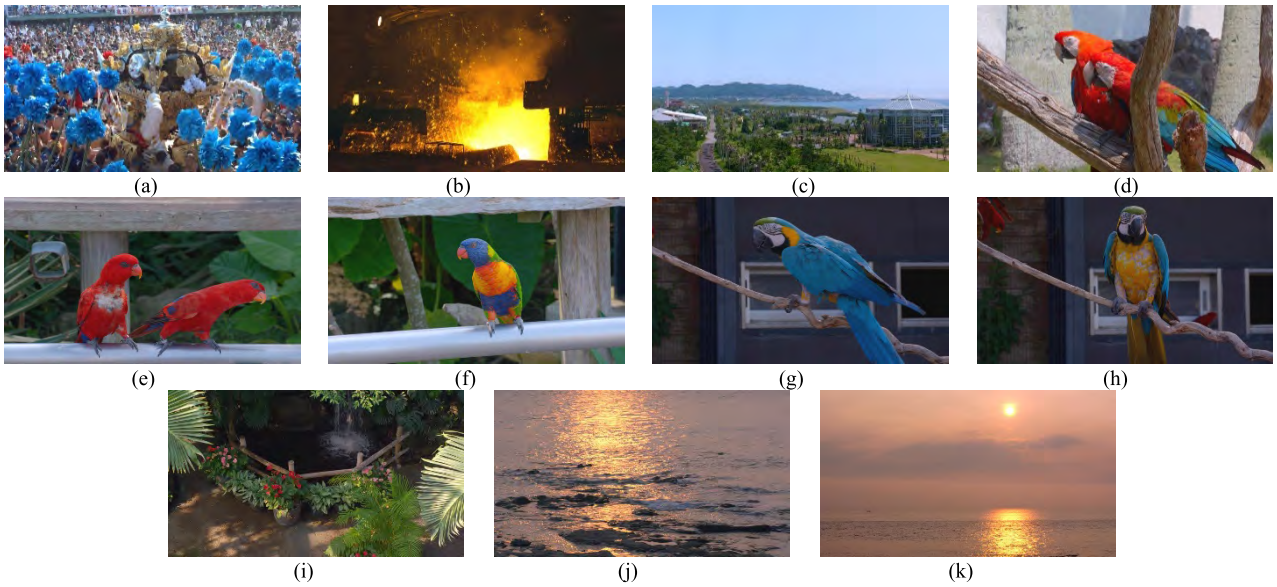


FIGURE 2. Test Images. (a) *Festival*. (b) *Steel Plant*. (c) *Coast*. (d) *Bird 1*. (e) *Bird 2*. (f) *Bird 3*. (g) *Bird 4*. (h) *Bird 5*. (i) *Greens*. (j) *Surface*. (k) *Sunset*.

We identify the relationship between the chromaticities in the original BT.2020 gamut image and the chroma-clipped the BT.2020 gamut image. The first method anisotropically diffuses the BT.709 chromaticities according to the direction of the original chromaticities in the BT.2020 color space. We also show a method to restore color information using CNN. Using these methods along with the BT.709 images gives the BT.2020 images with chromaticities out of the BT.709 color gamut.

There are three contributions from this study as follows.

1) We propose color space conversion method from BT.709 to BT.2020 using anisotropic chromaticity diffusion which generates chromaticities that the original BT.709 image does not possess while conserving the chromaticities inside BT.709 gamut.
2) We propose color space conversion method from BT.709 to BT.2020 using CNN. We consider two networks for color gamut extension. The one directly learns end-to-end mappings between BT.709 and BT.2020 images. The other only learns the residue between the original image and the output image from TC method 13] which just maintains each XYZ value without a gamut-extension. By combining two CNNs and two integration blocks, we propose six methods for gamut-extension using CNN, then we evaluate these methods using mean peak signal-to-noise (PSNR) metric.
3) We evaluate our methods using 11 test images and compare our methods to the conventional methods in [13] and [35]. Then we confirm that output images from our methods are always superior to conventional methods in the point of PSNR against original BT.2020 images. On average, we also confirm that our CNN methods can achieve a 2.31-dB gain against the TC method [13].

Since our method can estimate chromaticities which lost in BT.709 gamut, our method can be applied to color gamut scalability video coding as inter-layer prediction.

This paper is organized as follows. Section II presents related work. Section III overviews our proposed system using anisotropic diffusion. Section IV details our proposed system using CNN. Section V shows the experimental results using our two methods. Finally, Section VI concludes this report and describes future work.

II. RELATED WORK

A. COLOR SPACE CONVERSION

Figure 2 shows the test images: *Festival* and *Steel Plant* from the Institute of Image Information and Technical Engineers (ITE) ultra-high definition / wide-color-gamut standard test sequences (Series A) [6]; *Coast*, *Bird 1-5*, *Greens*, *Surface,* and *Sunset* from the LUCORE UHDTV HDR standard test sequence set [7] published by IMAGICA. Furthermore, Fig. 3 (a) and (b) show the Commission Internationale de l'Eclairage (CIE) xy chromaticity diagrams of two test images from [6] and the gamuts of BT.2020 and BT.709. The outer horseshoe shape represents the entire range of possible chromaticities in the real world. Each triangle indicates the range of representable chromaticities of BT.709 and BT.2020. The images in Fig. 2 were recorded with the BT.2020 color gamut. In the case of image color conversion from BT.709 to BT.2020, with the BT.709 gamut, chromaticities outside the gamut are represented as the RGB signal intensity values below 0% or above 100%. Because these cannot be expressed with usable intensity values, they are clipped and replaced with RGB values of 0% or 100%.

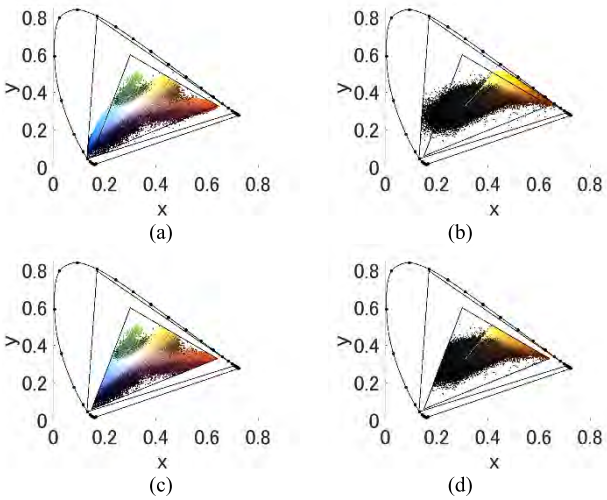


FIGURE 3. CIE xy chromaticity diagrams of the test images and gamut of BT.709 and BT.2020 Inner triangle: Gamut of BT.709. Outer triangle: Gamut of BT.2020. (a) *Festival* (BT.2020). (b) *Steel Plant*(BT.2020). (c) *Festival* (BT.709). (d) *Steel Plant*(BT.709).

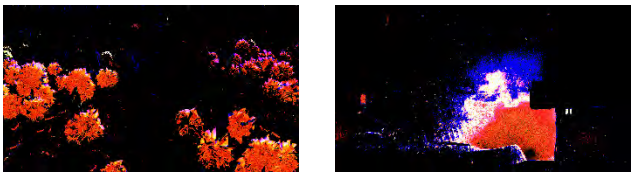


FIGURE 4. Differential image between the original BT.2020 and the chroma-clipped BT.2020 images. (a) *Festival*. (b) *Steel Plant*.

Although the technical report [8] standardized the linear conversion using a 3×3 transform matrix, it cannot be used to restore the chromaticities completely. Fig. 3 (c) and (d) show that chromaticity diagrams of two BT.709 (clipped) images. We can confirm that BT.709 images have only chromaticities within a BT.709 gamut triangle. we also show the differences between the original BT.2020 and the chroma-clipped BT.2020 images in Fig. 4. The clipped images have chromaticities within the BT.709 gamut due to gamut compression and clipping. They indicate the channel as well as how and where color information loss occurs through a linear conversion from BT.2020 to BT.709. For example, Fig. 4(a) shows that clipping occurs in the areas of blue fluffy things in Fig. 2(a) with an R channel.

Many researchers have reported mapping methods between different gamuts. These are referred to as gamut-extensions or gamut-compressions [9]. Gamut-extension (or gamut-expansion) achieves a conversion from a narrower gamut to a wider gamut, where the gamut-compression solves the inverse problem of gamut extension. Lee et al. proposed a gamut mapping method, which utilizes both lightness-mapping and multiple anchor points to print full-resolution color images on limited color output devices [10]. Anderson et al. proposed a gamut-expansion method using a multi-dimensional Look-up Table (LUT), which is trained with gamut-expanded images made by an artist from gamut-compressed images or is taken as truth data for a wider gamut [11]. Liu et al. proposed a hue-preserving gamut-expansion algorithm [12].

Laird et al. reported five gamut-extension algorithms [13]. The first, a true-color (TC) algorithm, maintains each XYZ value without a gamut-extension. It uses the simple conversion method describe in [6]. The second, a same drive signal (SDS) algorithm, is the simplest one. It performs a simple linear extension of the colors to make full use of the wider-color gamut, inducing both lightness and hue errors even for colors within a narrower gamut. The third to the fifth algorithms: a hybrid color mapping (HCM), a chroma adaptive, lightness-chroma adaptive use high-chroma-boost extension which is a saturation-dependent linear combination of TC and SDS algorithms. They successfully created chromaticities with an outer narrower-gamut while conserving some chromaticities in the inner narrower-gamut. However, not all colors are successfully preserved.

Louis et al. proposed Color Gamut Scalable Video Coding, which is a scalable extension of a High Efficiency Video Coding (HEVC) standard that supports different color gamuts in an enhancement and the base layer [14]. Their method used a gamut-extension based on a conversion matrix and an offset for the inter-layer prediction. During the standardization stage of the Scalability extension of HEVC (SHVC), many models have been proposed. In [15], a piecewise linear model is used for gamut conversion, while in [16], [17], a Color Look-up Table (CLUT) model is employed.

Herein we propose novel conversion methods to cope with both the conservation and generation of chromaticities.

B. CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE RESTORATION

CNN was invented in the 1980's [18]. Today deep CNNs are widely used in image processing methods because they successfully classify images [19], [20]. Image restoration methods using deep CNNs have also been proposed. Jain et al. proposed a natural image denoising method using CNN [21]. Deep learning has also been applied to image super-resolution [22], [23]. Moreover, CNN-based HDR imaging has also been investigated [5], [24]–[26]. And more methods for colorization [27] and inpainting [28], [29].

However, we cannot find a previous work on gamut-extension using deep CNN.

III. PROPOSED METHOD USING ANISOTROPIC DIFFUSION

Before explaining the proposed method, the relationship between the chromaticities in the original BT.2020 gamut image and the chroma-clipped BT.2020 image are described (Fig. 5). The inner red cube indicates the representable range of BT.709, and the outer black parallel pipe represents the range of BT.2020. The position and volume of each sphere indicate the respective $RGB_{BT.709}$ value and frequency. $RGB_{BT.709}$ stands for the BT.709 RGB signal.

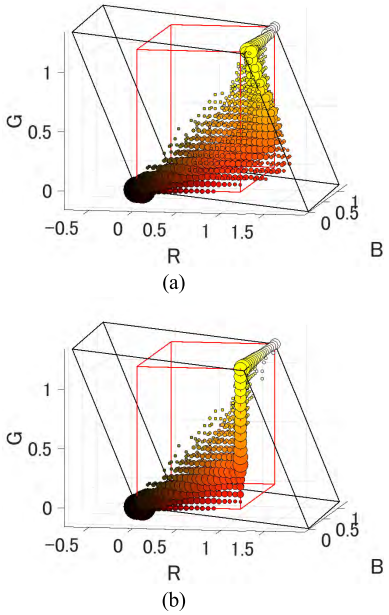


FIGURE 5. $RGB_{BT.709}$ histograms in *Festival.* **(a) Original BT.2020 image. (b) Chroma-clipped BT.2020 image.**

The chromaticities are condensed on the surface of the cube due to clipping (Fig. 5(b)). These results suggest that the chromaticity-diffusion process could be the inverse of chroma-clipping, and it appears as chromaticity-condensing.

However, even a chromaticity-diffusion process may be the inverse of chroma-clipping. As shown in Fig. 6, an isotropic diffusion kernel should not be used in the diffusion process. Considering the case where the chromaticities are diffused on the boundary of the BT.709 gamut, they should not diffuse inside of BT.709 because their original chromaticities should be on the outside.

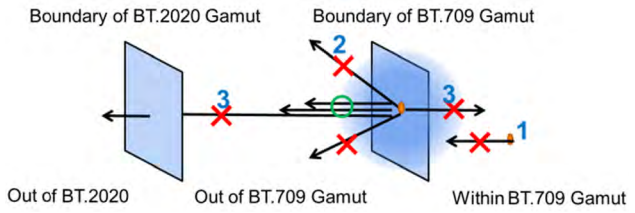


FIGURE 6. **Three policies of our method.**

The chromaticities should not be diffused beyond the boundary of the BT.2020 gamut. Hence, to keep non-saturated RGB values, some directions should not be diffused. That is, the chromaticities within BT.709 should not be moved to keep non-saturated RGB values. Because linear diffusion yields illegal chromaticity transitions, an anisotropic diffusion process needs to be devised to keep transitions legal.

Our method is based on three policies. The first is to not change the colors inside the BT.709 gamut. The second is to change clipped RGB values only. The third is to change colors within the BT.2020 gamut and outside the BT.709 gamut.

Figure 6 shows examples of transitions that violate these policies. Figure 7 describes the flow of our method. The first step yields the BT.709 image from the original BT.2020 image with a simple transformation and clipping. Then a chroma-clipped BT.2020 image is obtained using the TC algorithm in [13], which is a simple linear transformation [6]. This chroma-clipped BT.2020 image will be used as the input image for our method.

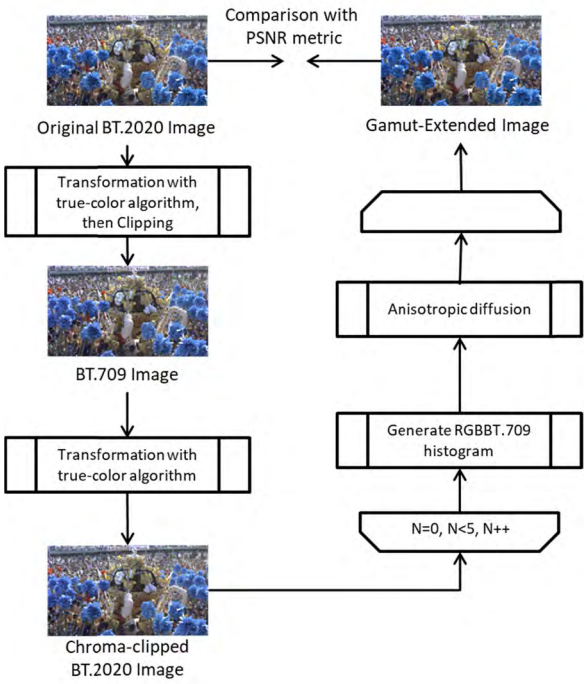


FIGURE 7. **Flowchart of our anisotropic diffusion method.**

In the first step, we obtain a $RGB_{BT.709}$ histogram similar to the one shown in Fig. 5(b). In the experiment described in the next section, we empirically use a 1% signal intensity as the histogram bin-width for each $RGB_{BT.709}$ axis.

To generate the chromaticities, which are of the BT.709 gamut only from the chromaticities within the BT.709 gamut, we diffuse the chromaticities anisotropically per the three aforementioned policies. According to the first policy, we diffuse only the chromaticities on the surface of the BT.709 cube. For each chromaticity on and outside the cube, an anisotropic kernel is generated based on the second and third policies.

Figure 8 shows an example of generating an anisotropic $25 \times 25 \times 25$ Gaussian kernel. In the experiment, we empirically use 0.001 for its parameter sigma. Then to prevent diffusing into illegal chromaticities, it is masked based on the second and third policies. After masking, the kernel is normalized to yield an anisotropic one for the target position, which then is used for diffusion. This process is applied to all chromaticities on and outside the BT.709 cube.

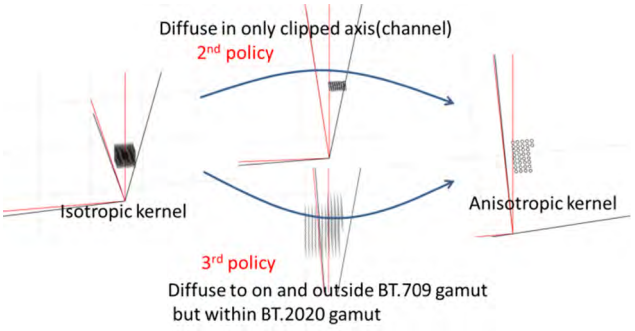


FIGURE 8. Example of generating an anisotropic kernel.

Generating an anisotropic kernel also can be described as $G(i, j, k, r, g, b)$ using the following equation, where (r, g, b)

$$G(i,j,k,r,g,b) = \frac{G'(i,j,k,r,g,b)}{\sum_i \sum_j \sum_k G'(i,j,k,r,g,b)} \tag{1}$$

$$G'(i,j,k,r,g,b) = M_{1R}(i,r) \cdot M_{1G}(j,g) \cdot M_{1B}(k,b) \cdot M_2(i,j,k,r,g,b) \cdot exp(-\frac{i^2+j^2+k^2}{2\sigma^2}) \tag{2}$$

$$M_{1R}(i,r) = \begin{cases} ((r+i) \le 0)?1:0, & r \le 0 \\ ((r+i) \ge MaxR)?1:0, & MaxR \le r \\ (i=0)?1:0, & otherwise \end{cases} \tag{3}$$

$$M_{1G}(j,g) = \begin{cases} ((g+j) \le 0)?1:0, & g \le 0 \\ ((g+j) \ge MaxG)?1:0, & MaxG \le g \\ (j=0)?1:0, & otherwise \end{cases} \tag{4}$$

$$M_{1B}(k,b) = \begin{cases} ((b+k) \le 0)?1:0, & b \le 0 \\ ((b+k) \ge MaxB)?1:0, & MaxB \le b \\ (k=0)?1:0, & otherwise \end{cases} \tag{5}$$

$$M_2(i,j,k,r,g,b) = !IsInside(r+i, g+j, b+k, BT.709) \cdot IsInside(r+i, g+j, b+k, BT.2020) \tag{6}$$

is the $RGB_{BT.709}$ of target chromaticity. i, j, and k are the variables for each axis. MaxR, MaxG, and MaxB are the maximum intensities for each channel in BT.709. $M_{1R}(i, r)$, M_{1G}(j, (g), and $M_{1B}(k, b)$ are the mask functions to create the restrictions from the second policy. $M_2(i, j, k, r, g, b)$ is the mask function representing the restrictions based from the third policy. The function IsInside $(r, g, b, Gamut)$ returns the $RGB_{BT.709}$ values (r, g, b) inside $Gamut$(1) or not(0). ! (Exclamation) is the logical NOT operator. ? (Question) is the conditional ternary operator. $G(i, j, k, r, g, b)$ is the normalized function of $G'(i, j, k, r, g, b)$, and $G'(i, j, k, r, g, b)$ stands for a two-dimensional Gaussian function masked with multiple functions: $M_{1R}(i, r)$; M_{1G}(j, g); $M_{1B}(k, b)$ and $M_2(i, j, k, r, g, b)$.

Figure 9 shows the relationship among i, r, and $M_{1R}(i, r)$. It shows that the proposed method diffuses the chromaticities on target r within the white area. If target r is out of the range $(0, MaxR)$, $M_{1R}(i, r)$ returns 1 only when $(r+i)$ locates out of the range. If target r is within the range, $M_{1R}(i, r)$, it returns 0 only when i is zero. Hence, our method does not diffuse the chromaticities within the range. $M_{1G}(j, g)$ and $M_{1B}(k, b)$ are mask functions that refer to the G and B axes, respectively, and they have the same design as $M_{1R}(i, r)$.

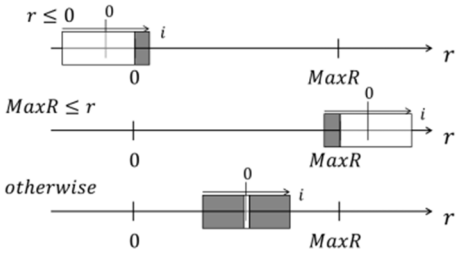


FIGURE 9. Explanation of the mask function $M_{1R}(i, r)$. black: 0, white: 1.

To promote diffusion of chromaticities, we performed the previous process iteratively. In our experiment, we iterate both the $RGB_{BT.709}$ histogram generation and the anisotropic diffusion explained in Sections 2.1 and 2.2 up to five times.

IV. PROPOSED METHOD USING CNN

We also propose another color gamut extension method using CNN (Fig. 10). In this chart, "w/o CNN" Image stands for the generated image using only with TC algorithm in [13], which is a simple linear transformation. (CNN1)–(CNN6) are our proposed methods using CNN in combination with two trained CNN models and two integration methods.

A. CNN MODELS FOR OUR METHODS

We considered two ways to perform the gamut-extension using CNN. In the first one, CNN(A), the RGB value of the input image is $RGB_{BT.709}$ and that of the output is $RGB_{BT.2020}$. In other words, CNN(A) learns an end-to-end mapping between the $RGB_{BT.709}$ images and the $RGB_{BT.2020}$ images. That means that CNN learns the linear relationship between $RGB_{BT.709}$ and $RGB_{BT.2020}$, which is already known as the conversion matrix, as well as how to restore the saturated areas in $RGB_{BT.709}$ images. In the other one, CNN(B), the input is a chroma-clipped BT.2020 image. CNN(B) learns an end-to-end mapping between chroma-clipped $RGB_{BT.2020}$ images and ideal $RGB_{BT.2020}$ images. Hence, CNN(B) only learns how to restore saturated signals.

Although they have different purposes, both CNN(A) and CNN(B) use the same network structure. Figure 11 shows our three-layer CNN architecture. We adopted a shallow network, not deep one. According to Dong et al. [23], in the image super-resolution method using CNN, they concluded that using a deeper layer makes it harder to set the appropriate learning rates to guarantee convergence, since their CNN network contains no pooling layer or full-connected layer,

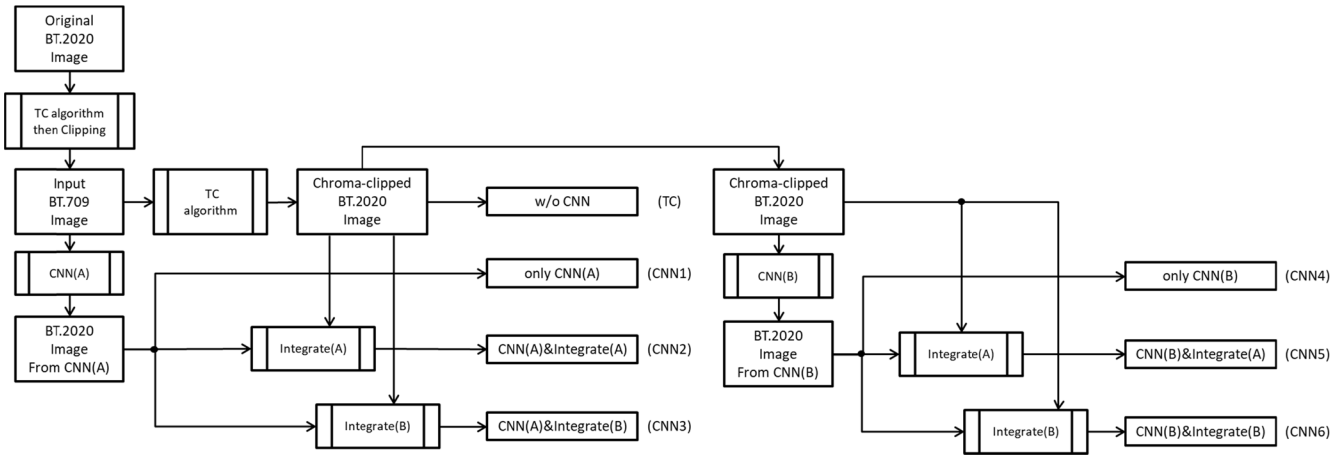


FIGURE 10. Overview of our CNN method.

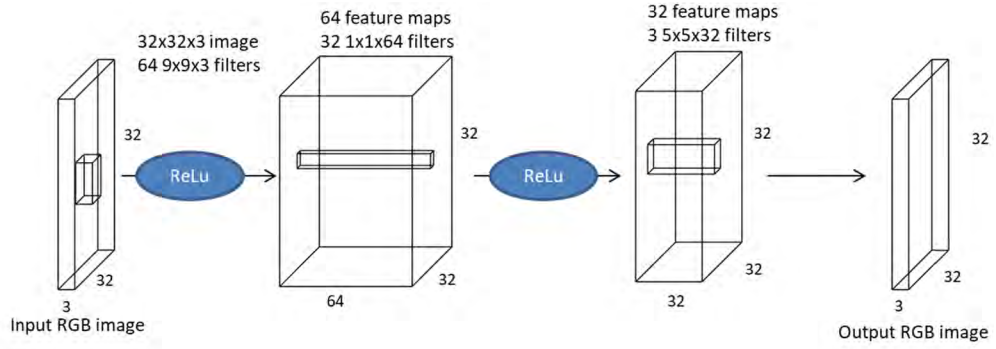


FIGURE 11. Our CNN architecture.

this it is sensitive to the initialization parameters and learning rate. Even if it converges, the network may fall into a bad local minimum. Consequently, the learned filters are less diverse even when given sufficient training time. Since the training purpose of our network is quite similar to that of their method, we followed their idea about the CNN architecture. As mentioned later in this section, we also confirmed that deeper network did not result in better.

Learning the end-to-end gamut-extension function F requires network parameters $\Theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$ to be estimated, where W_k represents the filters for k-th layer and B_k represents the biases for k-th layer. This is achieved by minimizing the loss between the estimated gamut-extended BT.2020 images $F(\boldsymbol{Y}; \Theta)$ and the corresponding ground truth BT.2020 images $\mathbf{X}$. Given a set of Original BT.2020 images $\{\mathbf{X}_i\}$ and their corresponding input images $\{\mathbf{Y}_i\}$, we used the Mean Squared Error (MSE) as the loss function:

$$L(\Theta) = \frac{1}{N} \sum_{i=1}^{N} \|F(\boldsymbol{Y}_i; \Theta) - \boldsymbol{X}_i\|^2 \qquad (7)$$

where N is the number of training samples.

As mentioned above, the chromaticities within BT.709 should keep a non-saturated RGB value through the gamut-extension. Due to the shallowness of our CNN architecture, CNN may not always perform selective color-extension only in saturated areas. It may include non-saturated areas. To reduce these errors, we also considered correcting the non-saturated area in the generated images from CNNs by integrating with chroma-clipped images.

B. TWO INTEGRATION (CORRECTION) BLOCKS

The purpose of these correction blocks is simple. After generating the BT.2020 images with CNNs, the correction blocks replace their pixel values in a non-saturated area with the one of the Chroma-clipped BT.2020 image at the corresponding position. To keep the chromaticities within BT.709. The Integrate(A) block performs this replacement only. In addition to above, the Integrate(B) block validates the generated pixel values in the saturated area, independently of each RGB channel. In particular, if CNN generates an illegal pixel value in a saturated area, the block corrects the pixel value with the saturated value. For example, when checking the R channel pixel value, the CNN output pixel values in the over-saturated area must be greater than the over-saturated value. The ones in the under-saturated area must be less than the under-saturated

value. If the pixel values break these rules, then the block corrects these values to reduce error.

C. OUR SIX CNN METHODS

In order to evaluate each CNN and integration block, we propose six methods for gamut-extension using CNN (Fig. 10), by combining our two CNNs and two integration blocks then evaluate them to see that which one is the best combination. The ''only CNN(A)'' directly uses the output from CNN(A). The ''CNN(A)&Integrate(A)'' and ''CNN(A)&Integrate(B)'' are the results from each Integration block. The ''only CNN(B)'', ''CNN(B)&Integrate(A)'', and ''CNN(B)&Integrate(B)'' are also the same but they use CNN(B).

D. DATASET FOR TRAINING

To train CNN, we used the private video sequences provided by NHK STRL. The sequences were recorded by conforming to the BT.2020 gamut. They consisted of 89 scenes with an 8K (7680×4320) resolution and 12-bit depth per channel. Their length was about 21,711 seconds (1,302,660 frames).

The following steps were used to create the dataset. First, we decimated the frames to reduce similar frames. Because the decimation factor was 180, 7237 frames remained. Then we separated the 89 scenes into a training set and a test set. Here, we used ten scenes for the test set, which is about 10% of all the frames. All the other scenes were used for the training set. Furthermore, we divided all 7237 frames into 64 540p (960×540) images per frame, and then randomly held $1/320^{\text{th}}$ of them. Finally, we extracted 32×32×3 image patches from the remaining 540p images, where the patch strides were 16. The total number of patches for training was 2,418,528.

E. PRELIMINARY EXPERIMENTS

As preliminary experiments, we evaluated our six CNN methods against the simple TC algorithm by calculating PSNR between the generated images and the corresponding ground truths. We implemented their methods using Caffe [30] and MATLAB. We trained CNN (A) and (B) on a standard desktop with a GeForce GTX 1080 Ti with 12 GB VRAM and it took around eleven hours for each CNN. We tested using images from both the training and the test sets.

Firstly, we compared performance of solvers for training, ADAM [26], AdaGrad [31], RMSProp [32] and AdaDelta [33]. The parameters for ADAM $(\beta_1, \beta_2, \varepsilon)$ were $\left(0.9, 0.999, 10^{-8}\right)$. The parameter rms_dacay for RMSProp was 0.99. The parameter ε for AdaDelta was 10^{-6}. The number of iterations was one million for all solvers. In order to check the difference among outputs with solvers, we compared solvers using outputs from CNN4 method which means direct output from CNN(B). Figure 12 compares the mean PSNR of generated images with solvers. Figure 12 (a) shows the mean PSNR when only the test set images are used as inputs. Similarly, Figure 12 (b) shows the one when only the train set images are used and Figure 12 (c) shows the one when both sets are combined and used. Since the ADAM solver has the best performance as shown in Figure 12, we use the ADAM solver for further experiments.

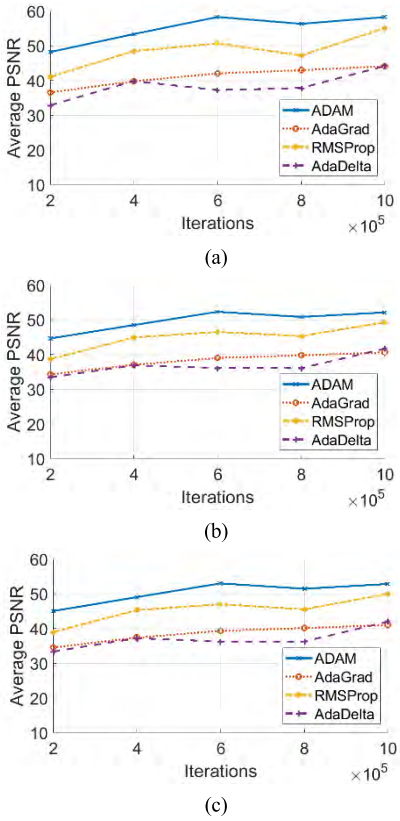


FIGURE 12. Mean PSNR with each solver. (a) Only test set. (b) Only train set. (c) Test and train set.

Secondly, we also compared performance among networks which have different the number of layers and the one of filters. Figure 13 compares the mean PSNR of generated images with networks. The network named ''3layers'' is the same as the network shown as Figure 11. Its filter sizes for each layer are 9, 1, and 5 so, it could also be denoted as 9-1-5. The two networks named ''4layers'' has 9-1-1-5 network structure which is added the third layer on 9-1-5 network. The additional layer consists of 1x1 convolutional layer and ReLU layer. The number of filters in the additional layer for ''4layers(A)'' was 16. The one for ''4layers(B)'' was 32. Similarly, ''5layers'' has 9-1-1-1-5 network. For ''5layers'', The one in the third layer was 32 and the one in the fourth layer was 16. As shown in Figure 13, deeper networks do not result in better performance. The similar results are shown also in the study for super resolution using CNN [23]. According to these results., we use three-layer-network for further experiments.

Finally, we compared performance among our six CNN methods. Table 1 compares the mean PSNR of generated images from the TC algorithm and our six CNN methods. The method indexes correspond to Fig. 10. The values in

TABLE 1. Mean PSNR for each datasets [dB].

	TC	CNN1	CNN2	CNN3	CNN4	CNN5	CNN6
Test Set	58.29	53.04	58.81	**58.89**	55.97	58.68	58.79
Train Set	51.15	48.44	52.68	52.78	50.98	52.92	**52.96**
All	51.97	48.97	53.39	53.48	51.55	53.58	**53.63**

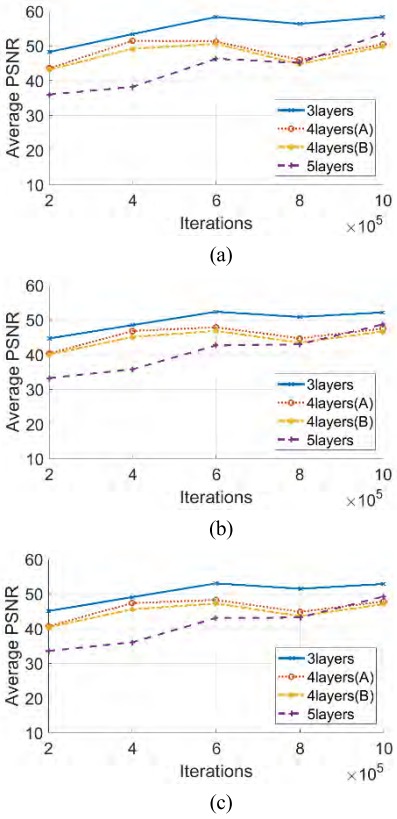


FIGURE 13. Mean PSNR with each network. (a) Only test set. (b) Only train set. (c) Test and train set.

the ''Test set'' and ''Train set'' rows show the mean PSNR when the test set images and train set images are used as inputs, respectively. ''All'' shows the results when both sets are combined and used. The direct CNN outputs contain more errors, as shown in columns (1) and (4). However, as shown in columns (2), (3), (5), and (6), the appropriate correction produces better results compared to the TC algorithm. According to these results, we made CNN3 and CNN6 methods representatives for comparison with other methods.

V. EXPERIMENTS

To evaluate our methods, we compared the PSNRs of the generated images from our methods to those from other methods. We used the test images shown in Fig. 2. We down-sampled those images into a 1920×1080 resolution. For all methods, we generated BT.709 images with the TC algorithm as the input. All the original, inputted, and generated images had 36 bit-per-pixel (12 bit-per-sample) bit-depth.

A. ANISOTROPIC DIFFUSION METHOD

This method was implemented on MATLAB. Then we conducted the experiments. First, we generated BT.2020 chroma-clipped images from the test images as inputs. Then we applied our methods to obtain the restored and gamut-expanded BT.2020 images. Table 2 shows the parameters for these experiments, which were empirically determined.

TABLE 2. Experimental parameters for diffusion.

Histogram resolution	0.01 $MaxRGB$ per bin
Sigma for Gaussian kernel	0.001
Gaussian kernel size	25×25×25

BT.2020, Input, and Generated (five times, iterated) images from *Festival*. Comparing Fig. 14(b) and Fig. 14(c) shows that the generated images have chromaticities outside the BT.709 gamut. Figure 15 shows the PSNR of the generated image from *Festival* against the ground truth. Since our anisotropic diffusion method independently processes image signal on each RGB channel, we separately indicated that plot for each RGB channel to make it easier to see the effect on each channel which has each image characteristic. Although the PSNRs can be improved against the input image (n = 0), this is not always the case. Figures 16 and 17 show the differential images between the original and input images or the generated images for *Festival* B and R channels. Gray indicates areas with values close to the original ones. Bright and dark areas denote values greater and less than the original ones, respectively. Channel B of *Festival* improves with the iterations (Fig. 15). This is confirmed in Fig. 16(a)–16(c). The dark areas decrease slightly as the iterations increase.

The R channel of *Festival* becomes worse (Fig. 15). Comparing Fig. 17(a) and Fig. 17(b) shows that the gray areas increase, while the dark areas spawn everywhere (Fig. 17(c)). Our method may cause over-diffusion beyond the correct chromaticities, resulting in a worse PSNR.

B. CNN METHODS

We also applied our CNN methods to test images and compared the generated images with the ones from the other methods, including TC, SDS, and HCM algorithms in [13],

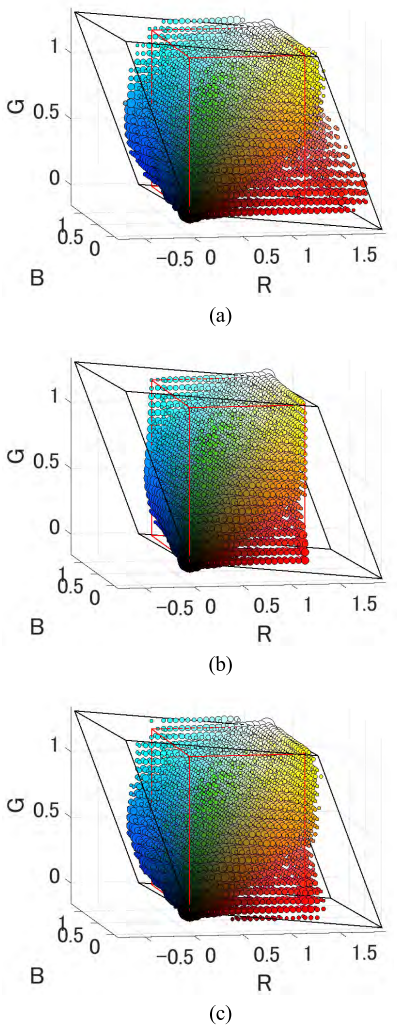


FIGURE 14. $RGB_{BT.709}$ histograms in *Festival.* (a) Original BT.2020 image. (b) Input(n = 0, TC Algorithm). (c) Generated (n = 5).

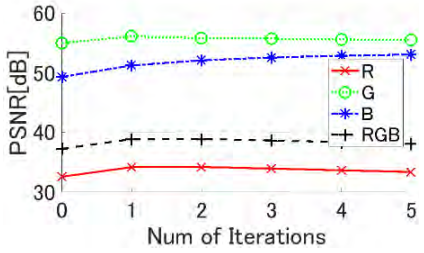


FIGURE 15. PSNR of the generated image from *Festival* against The Ground Truth.

the Xu's method [35] and our anisotropic diffusion method, then evaluated with PSNR, Structural SIMilarity (SSIM) [36] and Multi-Scale SSIM (MS-SSIM) [37] metrics.

Table 3 shows the PSNRs of the generated images from each method against each ground truth. ANI1 stands for our anisotropic diffusion method with only one pass (iteration), whereas ANI5 stands for the one with five iterations.

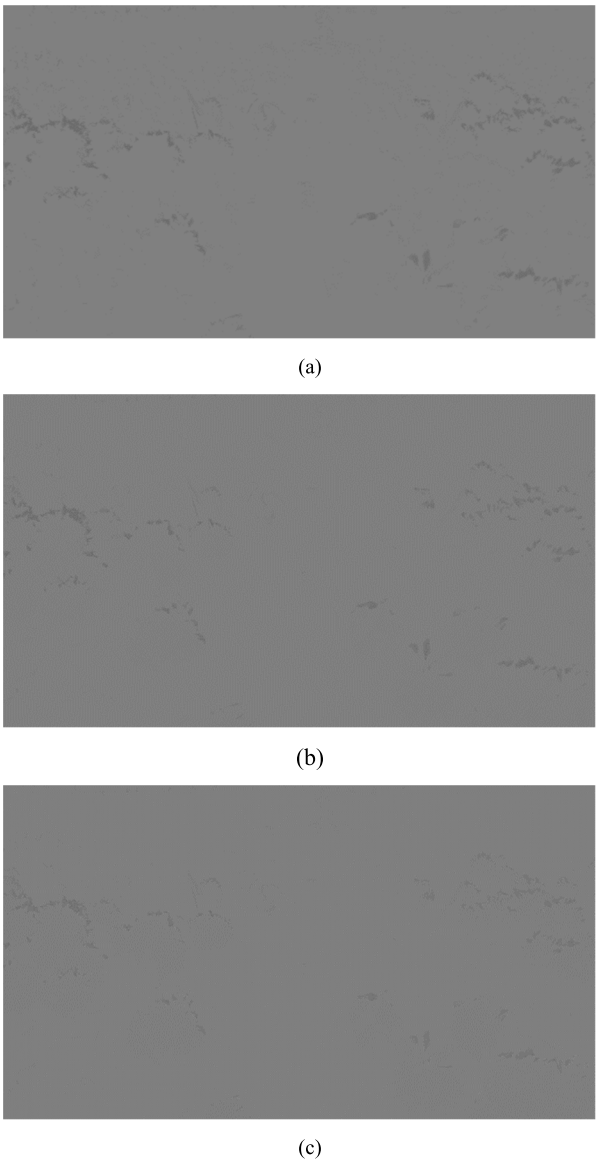


FIGURE 16. Differential images between original BT.2020 and input or generated images on *Festival* B channel. (a) Input (n = 0, TC Algorithm). (b) Generated (n = 1). (c) Generated (n = 5).

According to Table 3, of the 11 test images, our CNN method (CNN6) provides the best results for eight images. Our anisotropic diffusion method ANI5 provides the best for the remaining images. Since the TC method conserves the chromaticities inside BT.709, the calculated PSNRs only for the non-saturated area in the BT.709 images are extremely high values like around 60–70 dB. Due to quantization error in the gamut conversion, the values are not infinity. By contrast, the TC method does not affect the saturated area in the BT.709 images, but both the ANI5 and CNN6 methods induce changes while conserving the chromaticities inside BT.709. However, the TC method generates a better image than the ANI5 methods. On the other hand, our CNN6 method achieves a 2.31-dB gain against the TC method on average.

TABLE 3. PSNRs of generated images against ground truth [dB].

	Existing Work [13][36]				Diffusion		CNN	
	TC	SDS	HCM	Xu	ANI1	ANI5	CNN3	CNN6
Festival	37.24 (26.38)	24.75 (15.66)	31.29 (20.85)	27.48 (24.19)	38.85 (27.99)	38.08 (27.22)	39.08 (28.23)	**39.33** (**28.49**)
Steel Plant	39.52 (29.27)	26.82 (21.09)	33.12 (25.06)	29.37 (26.52)	39.57 (29.30)	37.79 (27.50)	39.90 (29.64)	**41.59** (**31.37**)
Coast	43.02 (35.13)	29.77 (37.78)	43.01 (35.20)	37.98 (**38.90**)	45.54 (37.65)	**46.71** (38.82)	43.20 (35.32)	44.62 (36.74)
Bird 1	44.18 (29.72)	29.80 (22.04)	38.31 (27.68)	31.43 (24.00)	44.61 (30.14)	43.92 (29.40)	44.51 (30.07)	**45.08** (**30.67**)
Bird 2	51.26 (37.43)	29.86 (28.94)	42.92 (35.89)	31.83 (32.39)	48.37 (34.55)	47.75 (33.92)	51.60 (37.76)	**52.69** (**38.87**)
Bird 3	40.28 (29.06)	31.79 (28.08)	41.90 (**33.63**)	34.49 (29.84)	41.95 (30.74)	42.07 (30.86)	42.10 (30.89)	**44.07** (32.86)
Bird 4	47.79 (36.97)	31.51 (23.72)	43.31 (33.65)	35.09 (33.83)	46.83 (36.02)	41.30 (30.48)	48.51 (37.69)	**51.56** (**40.77**)
Bird 5	47.29 (35.65)	36.54 (33.60)	47.36 (37.44)	37.94 (33.14)	50.07 (38.43)	49.14 (37.50)	47.77 (36.12)	**53.25** (**41.66**)
Greens	51.11 (38.78)	34.29 (35.61)	49.86 (39.47)	37.38 (34.93)	52.02 (39.69)	51.75 (39.42)	52.33 (40.00)	**54.43** (**42.12**)
Surface	39.46 (24.65)	31.38 (27.43)	39.45 (24.65)	31.93 (**29.79**)	40.02 (25.20)	**40.79** (25.98)	39.51 (24.70)	39.73 (24.92)
Sunset	43.34 (26.29)	28.09 (25.65)	42.67 (26.16)	28.61 (**28.86**)	43.97 (26.92)	**44.61** (27.55)	43.40 (26.35)	43.59 (26.54)
Average	44.05 (31.76)	30.42 (27.24)	41.20 (30.88)	33.05 (30.58)	44.71 (32.42)	43.99 (31.70)	44.72 (32.43)	**46.36** (**34.09**)

Figures between brackets indicate the PSNRs only in saturated area.

TABLE 4. SSIMs of generated images against ground truth.

	Existing Work [13][36]				Diffusion		CNN	
	TC	SDS	HCM	Xu	ANI1	ANI5	CNN3	CNN6
Festival	0.9979	0.9515	0.9930	0.9809	0.9984	0.9973	0.9985	**0.9989**
Steel Plant	0.9993	0.9582	0.9968	0.9865	0.9992	0.9989	0.9995	**0.9996**
Coast	0.9982	0.9624	0.9982	0.9948	0.9988	**0.9988**	0.9983	0.9988
Bird 1	0.9995	0.9668	0.9985	0.9894	0.9995	0.9993	0.9995	**0.9996**
Bird 2	0.9984	0.9441	0.9976	0.9828	0.9982	0.9980	0.9984	**0.9988**
Bird 3	0.9957	0.9512	0.9960	0.9822	0.9949	0.9938	0.9962	**0.9971**
Bird 4	0.9990	0.9655	0.9981	0.9803	0.9987	0.9968	0.9990	**0.9994**
Bird 5	0.9987	0.9727	0.9987	0.9820	0.9990	0.9989	0.9987	**0.9992**
Greens	0.9990	0.9718	0.9989	0.9825	0.9992	0.9992	0.9991	**0.9994**
Surface	0.9982	0.9567	0.9982	0.9909	0.9984	**0.9987**	0.9982	0.9983
Sunset	0.9995	0.9355	0.9994	0.9930	0.9996	**0.9996**	0.9995	0.9995
Average	0.9985	0.9579	0.9976	0.9859	0.9985	0.9981	0.9986	**0.9990**

TABLE 5. MS-SSIMs of generated images against ground truth.

	Existing Work [13][36]				Diffusion		CNN	
	TC	SDS	HCM	Xu	ANI1	ANI5	CNN3	CNN6
Festival	0.9964	0.9569	0.9826	0.9704	0.9963	0.9942	0.9972	**0.9973**
Steel Plant	0.9961	0.9690	0.9892	0.9782	0.9945	0.9927	0.9960	**0.9963**
Coast	0.9993	0.9910	0.9993	0.9971	0.9992	0.9990	0.9994	**0.9995**
Bird 1	0.9984	0.9853	0.9942	0.9893	0.9984	0.9981	0.9985	**0.9986**
Bird 2	0.9991	0.9830	0.9965	0.9902	0.9986	0.9984	0.9992	**0.9993**
Bird 3	0.9947	0.9844	0.9961	0.9902	0.9956	0.9954	0.9959	**0.9969**
Bird 4	0.9988	0.9840	0.9966	0.9905	0.9981	0.9952	0.9989	**0.9992**
Bird 5	0.9989	0.9899	0.9986	0.9933	0.9993	0.9991	0.9990	**0.9996**
Greens	0.9996	0.9900	0.9995	0.9948	0.9997	0.9997	0.9997	**0.9998**
Surface	0.9984	0.9885	0.9984	0.9909	0.9986	**0.9988**	0.9984	0.9985
Sunset	0.9985	0.9918	0.9984	0.9932	0.9986	**0.9987**	0.9985	0.9985
Average	0.9980	0.9831	0.9954	0.9889	0.9979	0.9972	0.9982	**0.9985**

As described in previous subsection, ANI5 does not always perform better than doing nothing. Our CNN6 method always wins against the TC method for all 11 test images. In Table 3, Figures between brackets indicate the PSNRs calculated only for the saturated areas in the BT.709 images. According to the result for *Bird 3*, although our CNN6 method does

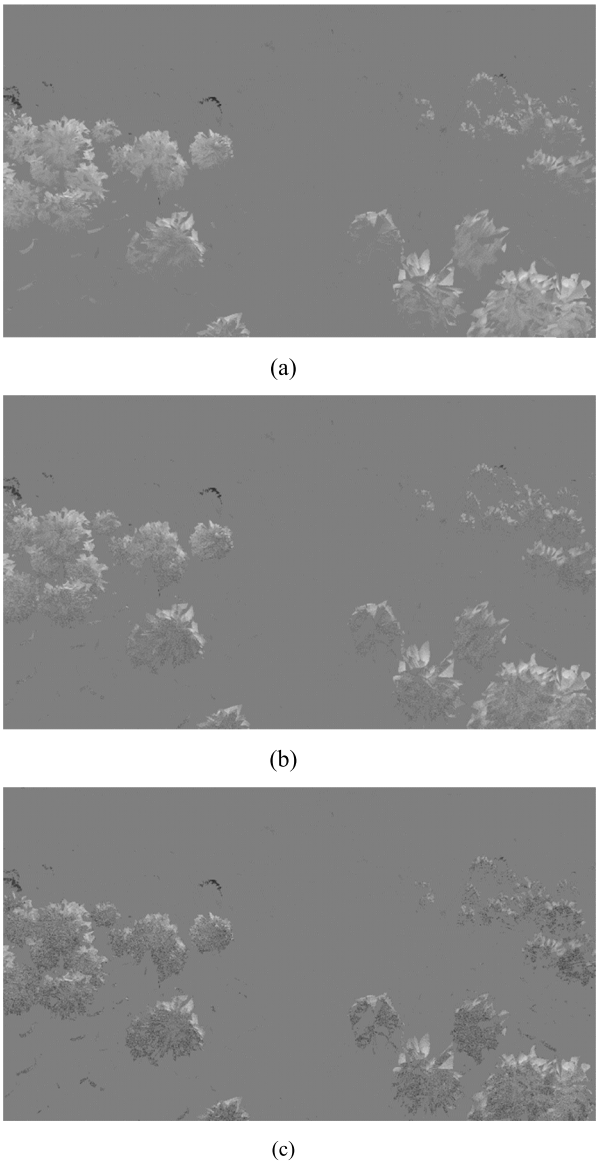


FIGURE 17. Differential images between original BT.2020 and input or generated images on *Festival* R channel. (a) Generated (n = 1). (b) Generated (n = 5). (c) Generated (n=5).

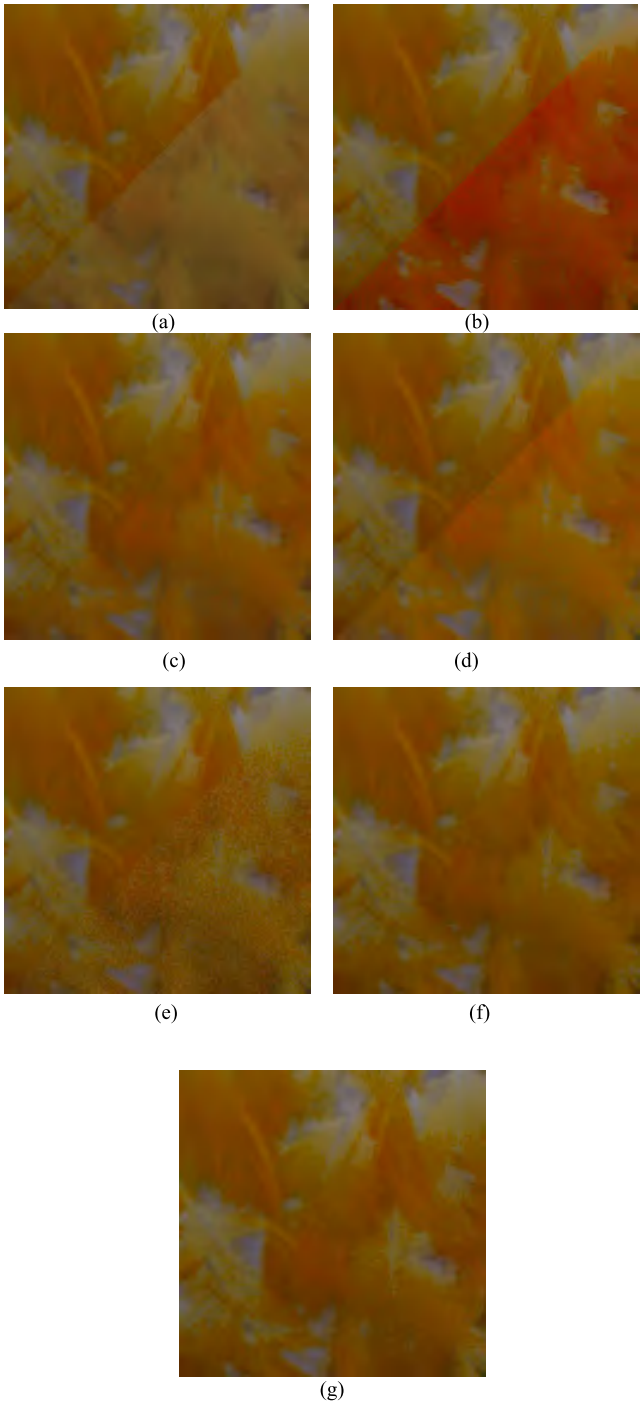


FIGURE 18. Comparisons between the ground truth and generated images from each method at a part of *Bird 5*. (a) The TC method. (b) The SDS method. (c) The HCM method. (d) The Xu method. (e) Our ANI5 method. (f) Our CNN6 method. (g) The ground truth. Upper left half: The ground truth. Lower right half: Generated images.

not give the best PSNR in Table 4, which is only for the saturated area, it does in Table 3, which is for the whole picture. This means that the HCM method, which is the best method for *Bird 3* in Table 4, does not perfectly conserve the chromaticities inside BT.709. Hence, the PSNR in HCM drops in the non-saturated area but that in our CNN6 method does not. This is why our CNN6 method gives the best result for the whole *Bird 3* image. Our CNN6 method can achieve a 2.33-dB gain against the TC method on average when only the saturated area is calculated. That means our CNN6 method can selectively and properly restore chromaticities.

Table 4 and 5 shows the SSIMs and the MS-SSIMs of the generated images from each method against each ground truth. As the same as the results with the PSNR shown in Table 3, our diffusion method ANI5 and CNN method CNN6 were the best methods. However, the gains were not so significant.

We also subjectively checked the generated images from each method on an LG OLED55C6P OLED TV, which supports the BT.2020 color gamut. Although the SDS method and the HCM method perform over color enhancements in the

whole area, our diffusion and CNN method maintain the color except in the saturated area.

Figure 18 compares the ground truth and the generated images from each method at a part of *Bird 5*. The upper left halves of each image are the ground truth. The lower right ones are the generated images from each method. The TC method loses color intensity (Fig. 18(a)), confirming the diagonal boundary. The SDS and the Xu method performs an over color enhancement (Fig. 18(b), 18(d)). The HCM method also performs a smaller color enhancement compared to the SDS method. Regardless, we can still confirm the boundary, as shown in Fig. 18(c). Although our ANI5 method gives a better image than the one from the TC method in the point of PSNR, the generated image from our ANI5 method is noisy (Fig 18. (e)). This is because our ANI5 method spatial-independently diffuses the chromaticity without considering the spatial correlations. In the combined image of the ground truth and generated image from CNN6 method, we can hardly see boundary (Fig. 18(f)). This means that our CNN6 method can generate an image similar to the ground truth.

VI. CONCLUSION AND FUTURE WORK

Herein we propose two color space conversion methods from BT.709 to BT.2020 using anisotropic chromaticity diffusion and CNN. Both of these methods can provide BT.2020 images with chromaticities out of the BT.709 color gamut that the original BT.709 did not possess. We also compare our methods to the conventional methods in [13]. One of our CNN methods can achieve a 2.31-dB gain against the TC method on average.

In the future, we will try to also utilize spatial–temporal correlations in an anisotropic diffusion method and to consider suitable termination criterion for iteration. We also will try to combine our CNN methods with the method for inverse tone mapping [5] as well as apply our methods to color gamut scalability video coding to improve inter-layer prediction. We also will try to apply Residual Network [34] to confirm the performance with much deeper network structure.

REFERENCES

[1] E. Nakasu, "Super hi-vision on the horizon: A future TV system that conveys an enhanced sense of reality and Presence," *IEEE Consum. Electron. Mag.*, vol. 1, no. 2, pp. 36–42, Apr. 2012.

[2] *Recommendation ITU-R BT.2020-2: Parameter Values for Ultra-High Definition Television Systems for Production and International Programme Exchange*, document ITU-R BT.2020-2, 2015.

[3] *Recommendation ITU-R BT.709-6: Parameter Values for the HDTV Standards for Production and International Programme Exchange*, document ITU-R BT.709-6, 2015.

[4] Y. Matsuo, R. Takada, S. Iwasaki, and J. Katto, "Image super-resolution method using registration of multi-scale wavelet components with consideration of digital cinema noise,"(in Japanese), *J. ITE*, vol. 68, no. 2, pp. J92–J98, 2014.

[5] K. Hirao, Z. Cheng, M. Takeuchi, and J. Katto, "Convolutional neural network based inverse tone mapping for high dynamic range display using LUCORE," in *Proc. IEEE Int. Conf. Consum. Electron.*, Jan. 2019, pp. 1–2.

[6] *The Institute of Image Information and Television Engineers, Ultra-High Definition/Wide-Color-Gamut Standard Test Sequences—Series A*. Accessed: Jun. 23, 2019. [Online]. Available: http://www.ite.or.jp/content/test-materials/uhdtv_a

[7] *LUCORE UHDTV HDR Standard Test Sequence Set*, IMAGICA Corp., Tokyo, Japan, 2018.

[8] *Recommendation ITU-R BT.2087-0: Colour Conversion from Recommendation ITU-R BT.709 to Recommendation ITU-R BT.2020*, document ITU-R BT. 2087-0, 2015.

[9] J. Morovic, "To develop a universal gamut mapping algorithm," Ph.D. dissertation, Colour Imag. Inst., Univ. Derby, Derby, U.K., 1998.

[10] C. S. Lee, W. Y. Park, S. J. Cho, and Y. H. Ha, "Gamut mapping algorithm using lightness mapping and multiple anchor points for linear tone and maximum chroma reproduction," *J. Imag. Sci. Technol.*, vol. 45, no. 3, pp. 209–223, May 2001.

[11] H. Anderson, E. Garcia, and M. Gupta, "Gamut expansion for video and image sets," in *Proc. 14th Int. Conf. Image Anal. Process.*, Sep. 2007, pp. 188–191.

[12] Y. Liu, G. Song, and H. Li, "A hue-preserving gamut expansion algorithm in CIELUV color space for wide gamut displays," in *Proc. 3rd Int. Congr. Image Signal Process.*, Oct. 2010, pp. 2401–2404.

[13] J. Laird, R. Muijs, and J. Kuang, "Development and evaluation of gamut extension algorithms," *Color Res. Appl.*, vol. c, no. 6, pp. 443–451, Dec. 2009.

[14] L. Kerofsky, A. Segall, and S.-H. Kim, "Color gamut scalable video coding," in *Proc. Data Compress. Conf.*, Mar. 2013, pp. 211–220.

[15] C. Auyeung and K. Sato, *AHG14: Color Gamut Scalable Video Coding With Piecewise Linear Predictions and Shift-Offset Model*, document JCTVC-N0271, Jul./Aug. 2013.

[16] P. Bordes, P. Andrivon, P. Lopez, and F. Hiron, *AHG14: Color Gamut Scalable Video Coding Using 3D LUT: New Results*, document JCTVC-N0168, Jul./Aug. 2013.

[17] P. Bordes, P. Andrivon, and R. Zakizadeh, *AHG14: Color Gamut Scalable Video Coding Using 3D LUT*, document JCTVC-M0197, Apr. 2013.

[18] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 346–361.

[20] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[21] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 769–776.

[22] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 49–64.

[23] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2015.

[24] N. K. Kalantari and R. Ramamoorthi, "Deep high dynamic range imaging of dynamic scenes," *ACM Trans. Graph.*, vol. 36, no. 4, p. 144, 2017.

[25] Y. Endo, Y. Kanamori, and J. Mitani, "Deep reverse tone mapping," *ACM Trans. Graph.*, vol. 36, no. 6, 2017, Art. no. 177.

[26] G. Eilertsen, J. Kronander, G. Denes, R. K. Mantiuk, and J. Unger, "HDR image reconstruction from a single exposure using deep CNNs," *ACM Trans. Graph.*, vol. 36, no. 6, 2017, Art. no. 178.

[27] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Trans. Graph.*, vol. 35, no. 4, p. 110, Jul. 2016.

[28] D. Pathak, P. Kráhenbúhl, J. Donahue, T. Darrell, and A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2016, pp. 2536–2544.

[29] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, "High-resolution image inpainting using multi-scale neural patch synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6721–6729.

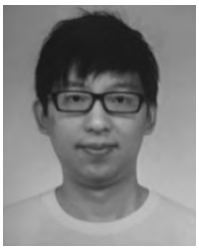
[30] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, Nov. 2014, pp. 675–678.

[31] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.
[32] T. Tieleman and G. Hinton, "RMSProp: Divide the gradient by a running average of its recent magnitude," COURSERA, Mountain Vie, CA, USA, Tech. Rep. 31, 2012.
[33] M. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*. [Online]. Available: https://arxiv.org/abs/1212.5701
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[35] L. Xu, B. Zhao, and M. R. Luo, "Color gamut mapping between small and large color gamuts: Part II. gamut extension," *Opt. Express*, vol. 26, no. 13, pp. 17335–17349, 2018.
[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[37] Z. Wang, E. P. Simoncelli, and A. C. Bovil, "Multi-scale structural similarity for image quality assessment," in *Proc. IEEE Conf. Signals Syst. Comput.*, vol. 2. Nov. 2003, pp. 1398–1402.

MASARU TAKEUCHI received the B.E. and M.E. degrees from Waseda University, Tokyo, Japan, in 2010 and 2012, respectively. He joined Sharp Corporation, in 2012. In 2015, he joined Waseda University. He is currently pursuing the Ph.D. degree with Waseda University and is a Research Associate. He is a member of IEICE.

YUSUKE SAKAMOTO received the B.E. from Waseda University, Tokyo, Japan, in 2017. He is currently pursuing the M.E. degree with Waseda University. He is a member of IEICE.

RYOTA YOKOYAMA received the B.E. from Waseda University, Tokyo, Japan, in 2018. He is currently pursuing with the M.E. degree with Waseda University. He is a member of IEICE.



HEMING SUN received the B.E. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2011, and M.E.'s from Waseda University and Shanghai Jiao Tong University, in 2012 and 2014, respectively, through a double-degree program. In 2017, he earned the Ph.D. degree in engineering from Waseda University, where he is currently an Assistant Professor. He was a Researcher with NEC Central Research Laboratories, from 2017 to 2018. His interests include algorithms and VLSI architectures for multimedia signal processing and neural networks.

YASUTAKA MATSUO received the M.E. from Nara Institute of Science and Technology (NAIST), Nara, Japan, in 2001, and the Ph.D. degree from Waseda University, Tokyo, Japan, in 2014. He joined Japan Broadcasting Corporation (NHK), in 2001, and NHK Science and Technology Research Laboratories, Tokyo, Japan, in 2004. He is a member of IEICE and ITE.

JIRO KATTO received the B.S., M.E., and Ph.D. degrees in electrical engineering from The University of Tokyo, in 1987, 1989, and 1992, respectively. He joined NEC, in 1992, and then joined Waseda University, in 1999. He is a Fellow of IEICE, and a member of ACM and IIEEJ.