REVISITING GAMUT EXPANSION FOR COLOR SPACE CONVERSION

HOANG MINH LE

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

YORK UNIVERSITY

TORONTO, ONTARIO

September 2024

Abstract

Cameras and image-editing software are capable of processing images in the wide-gamut ProPhoto color space, which encompasses 90% of all visible colors. However, when images are prepared for sharing, this rich color representation is transformed and clipped to fit within the smaller-gamut standard RGB (sRGB) color space, which represents only 30% of visible colors. Recovering the lost color information poses a challenge due to this clipping procedure. We propose three methods to address this issue. The first method proposes a deep neural network specifically trained for wide-gamut color restoration, utilizing datasets generated by a software-based camera image signal processor that produces pairs of ProPhoto and sRGB images. The second method implements a technique that incorporates a small set of color values sampled from the original ProPhoto image that is saved together with the final smaller-gamut sRGB image. The method then uses the subsampled wide-gamut color values to estimate the original ProPhoto image from the sRGB image. The third method proposes a lightweight multi-layer perceptron (MLP) trained on pairs of ground truth and clipped ProPhoto values during the gamut compression phase. The MLP is saved as metadata in the sRGB image and can later be used to predict and restore the original wide-gamut colors during the gamut expansion phase. Additionally, we have created several large-scale public datasets of wide-gamut/small-gamut image pairs to support research on color space conversion.

To my wife, parents, and aunts.

Acknowledgment

First, I would like to express my sincere gratitude to my advisor, Prof. Michael S. Brown, for his continuous support, patience, motivation, enthusiasm, and immense knowledge. When I first came to York University, I never expected to have such a great advisor. Richard Feynman once said: "Students don't need a perfect teacher. Students need a happy teacher, who's gonna make them excited to come to school and grow a love for learning." Fortunately, I have both. I have a perfect and happy teacher, from whom I have learned how to address problems in a scientific way and how to become an independent thinker. It is one of the best things I have ever learned.

Besides my advisor, I would like to thank the rest of my thesis committee: Kosta Derpanis and Marcus A. Brubaker, for their insightful comments and encouragement, and for the hard questions that allowed me to see my research from various perspectives.

I am also grateful to my fellow labmates and colleagues: Mahmoud Afifi, Abdullah Abuolaim, Abdelrahman Abdelhamed, Sai Tedla, Ian MacPherson, Trevor Canham, Beixuan Yang, Seonghyeon Nam, Javier Vazquez-Corral, Jason Yu, Harry Thasarathan, Matt Kowal, Amin Fadaeinejad, Ali Maleky, Rang Nguyen, Hue Nguyen, Niels Leif Bracher, Aditya Arora, Brian Price, Scott Cohen, Taehong Jeong, Hyun Joon Shin, Luxi Zhao, Abhijith Punnappurath, Lilian Bialokozowicz, Tristan Sylvain, Peter Allan Insley Forsyth, Vineel Nagisetty, Edward J. Smith, and Greg Mori, for their support, their stimulating

discussions, and all the fun we have had over the past few years..

My heartfelt thanks go to my friends: Anh Tan, Long Pham, Huy C. Le, Trung Luong, Thang Le, Cuong Lai, Rio Vu, Huy Nguyen, Sheena Ng, Chau Nguyen, Triet Hong, Nhat Le, Jake Nguyen. I am deeply grateful to all my friends, whose support, encouragement, and companionship have been invaluable throughout this journey, including those whose names are not listed here.

Finally, I wish to express my deepest appreciation to my family. To my parents, for giving birth to me, raising me, and providing me with a happy home and family from childhood to adulthood. Being your son is the greatest gift I have ever received. To my younger sister and my aunts, for their unwavering faith in my abilities and their constant encouragement. To my wife, Hoang Nguyen, for accepting me as who I am and growing old with me. Her endless support, understanding, and love have been a constant source of strength throughout this journey.

I am so lucky to have all of you in my life. I thank God every day for that.

Thank you to Canada; this country has been generous to me.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Problem Statement

This thesis is focused on the task of color space conversion. While it is well known that most color images are encoded using a weighted mixture of three primaries: red, green, and blue (i.e., RGB), it is often overlooked that the meaning of these red, green, and blue primaries differs depending on the color space used to encode the image. In particular, a color space explicitly defines the relationship of the color space's three primaries in terms of human vision. The span of these primaries defines what is known as the gamut of the color space. Different color spaces have been proposed depending on the task and targeted technology. For example, display and printing devices often have limits on their ability to generate visible colors. As a result, color spaces are designed to tightly bound the targeted technology's capabilities to maximize bit allocation. Color space conversion is the process of mapping between color spaces with different gamuts.

For example, the standard RGB (i.e., sRGB) color space was designed for the color display capabilities of 1990s cathode ray tube monitors and has been widely used. The color space was deemed optimal to encode colors that could be reproduced by display technology of the 1990s. Since sRGB's introduction, several wider gamut color spaces, such as Adobe RGB [2], Display P3[1], Ultra High Definition (UHD), and ProPhoto RGB [102],

[1]Display P3 was introduced by Apple and is based on the DCI-P3 color space [87].

have been proposed for use with improved display technology and image-editing software. For a simple comparison, the sRGB color space is capable of encoding around ∼35% of the colors visible to the human eye, while the medium-gamut spaces such as AdobeRGB, Display P3, and UHD are capable of representing approximately ∼50%, and the wide-gamut ProPhoto space encompasses an impressive ∼90% of all visible colors.

Despite the availability of these wider gamuts, most cameras and image editing software still encode images using the small-gamut standard RGB (sRGB) color space. This is due to its compatibility and historical prevalence, even though modern displays and software can support the broader ranges of newer color spaces.

One interesting thing is that most modern DSLR and smartphone cameras internally encode images using the ProPhoto color space [48]. Image-processing software, such as Adobe Photoshop, also uses this color-rich space to manipulate images, especially when processing camera RAW-DNG files. By processing images in the wide-gamut ProPhoto space, cameras and editing software offer users the flexibility to save an image in various color spaces—such as AdobeRGB, UHD, and Display-P3—that have much wider color gamuts than sRGB. However, these color spaces remain relatively unpopular, and most images are ultimately saved in sRGB. To convert color values between ProPhoto and sRGB, a gamut reduction step is applied that clips the wide-gamut color values to fit the smaller sRGB color gamut. Once gamut reduction is applied, it is challenging to recover the original wide-gamut values. As a result, when images are converted back to a wide-gamut color space for editing or display, much of the color fidelity is lost, as shown in Figure 1.1.

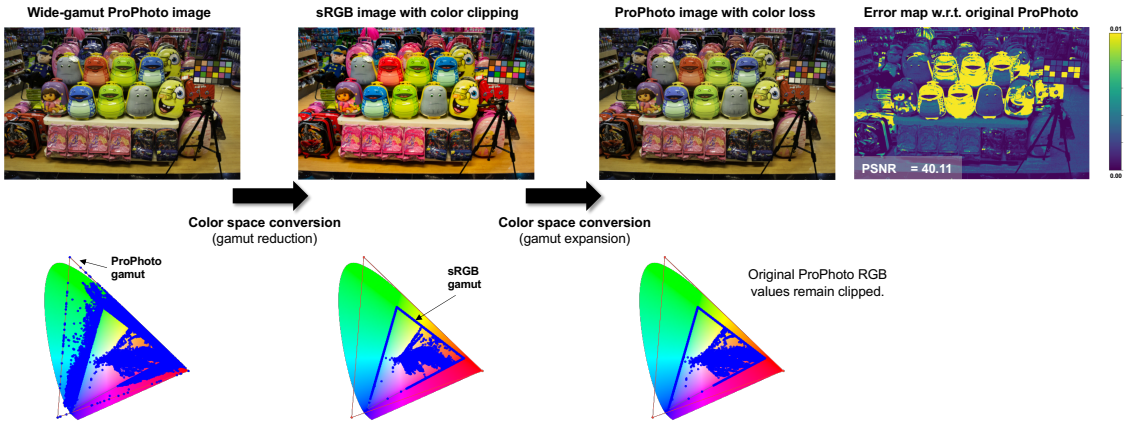


Figure 1.1: Consider an image encoded in a wide-gamut color space (ProPhoto). When the image is converted to a small-gamut color space (sRGB), color values are clipped to fit the smaller gamut. If the sRGB image is converted back to the wide-gamut color space, the clipped colors cannot be recovered, resulting in noticeable color loss.

1.2 Contributions

This dissertation mainly concerns gamut expansion as required during color space conversion. Our approach to this problem is to frame it as a restoration problem. One of the key insights used in our approaches is that cameras internally capture images in a wide-gamut color space (ProPhoto) before compressing and clipping the colors to fit sRGB's smaller gamut. This availability of the wide-gamut values before gamut reduction allows us to treat the problem as one of restoration, where the goal is to recover a known signal.

To tackle the problem of restoring wide-gamut colors based on the insight, this research explores three innovative methods. The first method involves creating a deep neural network tailored for wide-gamut color restoration, employing datasets produced by software-based camera image signal processors (ISPs) for training. Our second method proposes to include a small sub-sampling of the color values from the ProPhoto image state in the

camera to the final saved JPEG/HEIC image. We demonstrate that having this additional wide-gamut metadata available during color space conversion greatly assists in constructing a color mapping function to convert between color spaces. For the final one, inspired by neural implicit representations for 2D images, we propose a method, called *GamutMLP*, that optimizes a lightweight multi-layer-perceptron (MLP) model during the gamut reduction step to predict the clipped values. *GamutMLP* takes approximately 2 seconds to optimize and requires only 23 KB of storage. The small memory footprint allows our GamutMLP model to be saved as metadata in the sRGB image—the model can be extracted when needed to restore wide-gamut color values. Furthermore, we present several color gamut datasets consisting of wide-gamut/small-gamut image pairs, carefully selected and processed for the purpose of training and evaluating our methods.

1.3 Dissertation Outline

The remainder of this thesis is organized as follows.

- Chapter 2 - Preliminaries. This chapter introduces the fundamentals of color and color spaces, along with the differences between them. Furthermore, as our approach utilizes images captured in the wide-gamut ProPhoto color space, we will also share insights into the in-camera processing involved.

- Chapter 3 - Related works. This chapter examines existing studies on gamut reduction and gamut expansion. Following that, we offer a concise overview of related areas such as RAW recovery/derendering, image-to-image translation, and implicit neural representation.

- Chapter 4 - Gamut restoration via deep learning. This chapter focuses on our first approach, which involves designing a deep neural network tailored to restore wide-

gamut colors. The network is trained by employing datasets created by software-based camera image signal processors.

- Chapter 5 - Gamut restoration via image samples. This chapter focuses on our second approach, which captures a small sample of the color values from the camera's original ProPhoto image state and save them into the final, smaller-gamut sRGB image. This process assists in making the restoration effort more straightforward.

- Chapter 6 - Gamut restoration via MLP. This chapter focuses on our final approach, which creates an MLP that learns from pairs of ground-truth and clipped ProPhoto values during the gamut compression process. This MLP is designed to accurately predict and recover the original wide-gamut colors in the gamut expansion stage.

- Chapter 7 - Conclusion and future work. We begin by summarizing the key contributions of our work, highlighting the key features and advantages of our proposed approach. Next, we discuss the limitations and potential future directions for research in this domain. We also offer insights into the broader implications of our work for related fields and applications.

2 Preliminaries

To understand gamut manipulation, we will first discuss color and color spaces in this chapter. Gamut manipulation will be discussed in Chapter 3. Finally, since our method relies on images produced by cameras in the wide-gamut ProPhoto color space, details on in-camera processing will also be provided in this chapter.

2.1 Color

Color is not a physical property of an object but rather a perceptual sensation that arises from physical stimulation of light radiance. Human eyes biologically respond to the visible spectrum (light with wavelengths from 380 to 720 nm), giving us a sensation of color. For example, we perceive a 450-nm wavelength as "blue," 540 nm as "green," 650 nm as "red," etc. In the physical world, we rarely have a chance to see monochromatic light. Instead, objects reflect or emit a wide range of wavelengths that are denoted as a spectral power distribution (SPD). Our eyes respond to the incoming SPD to produce the sensation we refer to as color [26].

2.1.1 Establishing a device-independent color space

As previously mentioned, the physical world is composed of radiant energy. An SPD is often shown as a quantitative measurement of radiant energy, which can be light coming from

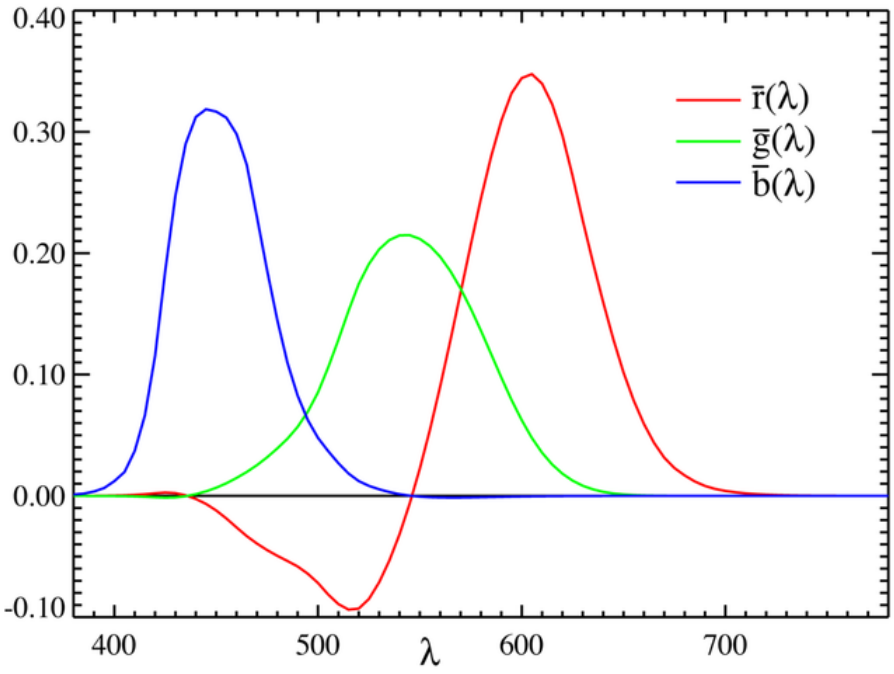


Figure 2.1: CIE RGB 2-degree Standard Observer. This plot illustrates how to mix coefficients of three primaries (red, green, and blue) to reproduce the corresponding monochromatic light at given wavelengths. Note: these functions are normalized so that the area of each curve is equal. Source: Wikipedia

a source (radiance) or light that falls on a surface (irradiance). The science of measuring such energy is so-called *radiometry*. The science of measuring the *perceived* radiant energy according to the sensitivity of the human eyes are known as *photometry* and *colorimetry*, which measures human perceived "brightness" and "color" respectively.

We briefly describe the experiments used to establish a device-independent color space that allows the mapping of radiant energy to perceived brightness and color. The first experiment is known as the "flicker photometry" experiment. Observers were asked to adjust the radiant power of a chromatic source light at a given wavelength to match a bright reference light with fixed radiant power. The results of this experiment enable us to

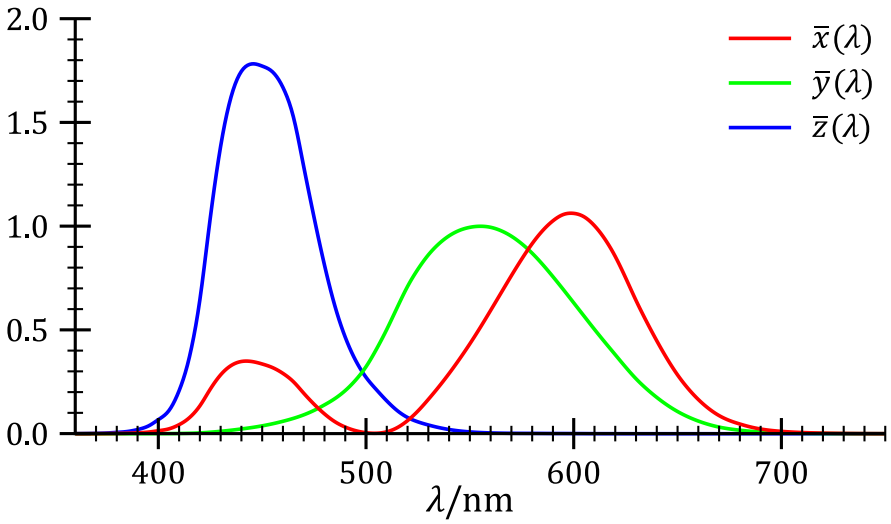


Figure 2.2: CIE 1931 XYZ 2-degree Standard Observer. This plot shows the coefficients $\bar{x}(\lambda), \bar{y}(\lambda), \bar{z}(\lambda)$ for the CIE 1931 XYZ. Source: Wikipedia

determine the perceived brightness of each wavelength. This experiment gave rise to the photopic luminosity function that can predict perceived brightness given an input SPD.

The second experiment to quantify all visible colors in terms of a standard set of primaries is the CIE RGB color matching experiment, which W. David Wright and John Guild carried out in the 1920s (CIE stands for the Commission Internationale d'Eclairage or International Commission on Illumination). This experiment is based on tristimulus color theory, stating that any source color could be matched by a linear combination of three independent primaries. In the experiment, several "standard observers" with no eye defects were asked to add or subtract three primaries so that a matched color should be the same as a given test color. The results of this experiment are shown in Figure 2.1, which gives us a way of how to combine the primaries to achieve the corresponding monochromatic light at a given wavelength. In some cases, since the three primaries could not reproduce the test colors, observers are asked to add primaries to the test colors, meaning that subtracting

primaries from the matched colors resulting in a negative region in Figure 2.1.

In 1931, the CIE defined a new canonical basis, called CIE 1931 XYZ, derived from CIE RGB data using a linear transformation. There are several desired properties of the CIE XYZ space over the CIE RGB space, such as the white point is defined at $X = 1/3, Y = 1/3, Z = 1/3$, Y is the luminosity function (brightness), and the XYZ basis are positive. Figure 2.2 shows the coefficients of CIE 1931 2-degree standard observer XYZ. We can describe SPDs in a canonical color space with CIE 1931 XYZ functions. For example, given an SPD, $S(\lambda)$, we have its corresponding mapped CIE 1931 XYZ values:

$$X = \int_{380}^{780} S(\lambda)\bar{x}(\lambda)\, d\lambda;$$
$$Y = \int_{380}^{780} S(\lambda)\bar{y}(\lambda)\, d\lambda;$$
$$Z = \int_{380}^{780} S(\lambda)\bar{z}(\lambda)\, d\lambda.$$

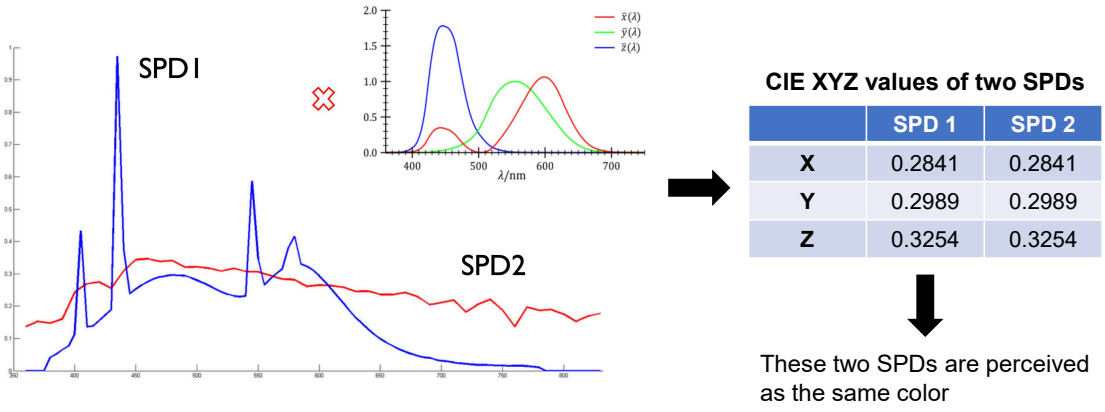


Figure 2.3: An example of two different SPDs mapping to the same values in CIE XYZ. They are perceived as the same color.

Using these CIE XYZ functions, we have a way to describe color quantitatively. For example, if two different SPDs map to the same CIE 1931 XYZ values, we should be considered the same perceived color (see Figure 2.3). CIE XYZ space is also called *device*

independent color space as values in this space are calculated based on human sensitivity to color, not specific to any device. Electronic devices such as cameras, scanners, printers, or displays have a mapping from their device values to the corresponding CIE XYZ values.

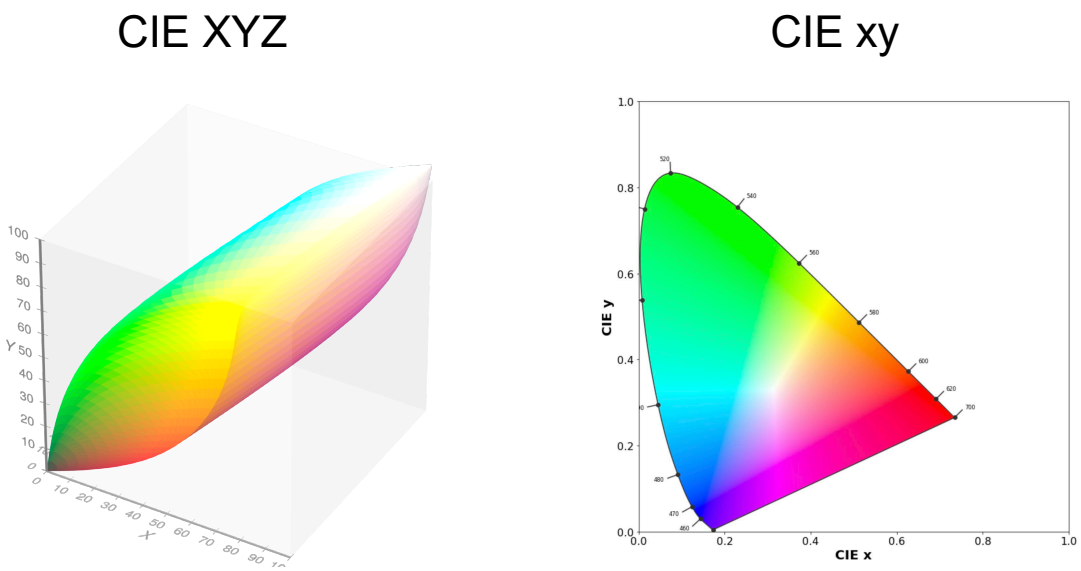


Figure 2.4: All visible colors in various device-independent color spaces CIE 1931 XYZ (left) and CIE xy chromaticity diagram (right).

Another quintessential representation of color is the luminance-chromaticity space. While CIE XYZ describes a color by a linear combination of three primaries, it is sometimes useful to investigate color separately in terms of luminance and chromaticity. In the luminance-chromaticity space, the chromaticity coordinates are called x and y, which can be derived from CIE XYZ as:

$$x = \frac{X}{X+Y+Z};$$
$$y = \frac{Y}{X+Y+Z}.$$

As previously described, the basis function Y from CIE XYZ is the luminance of the color, so we still keep Y as the luminance coordinate of this new color space, which is

called CIE xyY. Figure 2.4 illustrates all visible colors in 3D CIE 1931 XYZ (left) and their 2D projection as a horseshoe shape in a 2D plot with coordinates x and y from CIE xyY (right). This 2D plot is usually called the *CIE xy chromaticity diagram*. Note that there are many subsequent studies to improve the accuracy of the XYZ color space (e.g., CIE 1965 XYZ 10-degree standard observer); however, CIE 1931 XYZ and CIE 1931 xyY remain the dominant color spaces for visible color representation.

Another common device-independent color space is CIELAB or CIE $L^*a^*b^*$, defined by CIE in 1976. CIE $L^*a^*b^*$ was derived by transforming CIE XYZ values to make it more suitable for comparisons between colors. In particular, L^* is a perceptual brightness, a non-linear transformation from Y in CIE XYZ. The coordinates a^* and b^*, often ranging between -100 to +100, represent the four unique colors of human vision: red, green, yellow, and blue. The a^* values are related to red and green, while the b^* values are related to yellow and blue. The CIE $L^*a^*b^*$ is defined as more perceptually uniform than CIE XYZ, meaning that a numerical change in CIE $L^*a^*b^*$ will correspond to a perceived change in color, which may not be the case in CIE XYZ. Thanks to its relatively perceptually uniform characteristic, CIE uses CIE $L^*a^*b^*$ space for creating Euclidean measures of color difference, known as ΔE (Euclidean distance between two $L^*a^*b^*$ values).

CIE $L^*a^*b^*$ can be transformed to have a more cylindrical form as CIE $L^*C^*h^o$, which uses polar coordinates C^* and h^o:

$$C^* = \sqrt{a^{*2} + b^{*2}};$$
$$h^o = \arctan(\frac{b^*}{a^*})$$

The conversion to CIE $L^*a^*b^*$ if given the polar coordinates is:

$$a^* = C^* \cos h^o$$

$$b^* = C^* \sin h^o$$

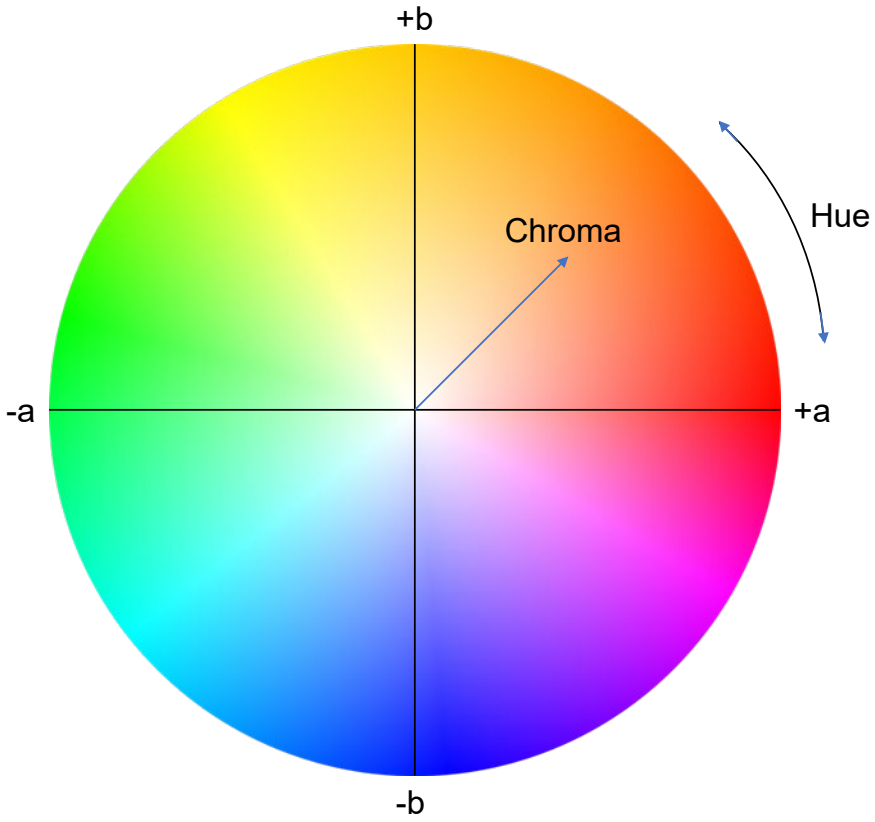


Figure 2.5: The relationship between $a*, b*$ and chroma, hue.

In CIE $L^*C^*h^o$, C^* represents chroma and h^o represents hue angle (angle of the hue in the CIE $L^*a^*b^*$ color wheel). Figure 2.5 shows the relationship between $a*, b*$ and chroma, hue. CIE $L^*C^*h^o$ is different from HSV or HSL color spaces, although they refers to saturation, color, and lightness. HSV or HSL is an alternative for RGB color spaces, which are not perceptually uniform like CIE $L^*C^*h^o$. CIE $L^*C^*h^o$ is also the most common color spaces on which gamut mapping algorithms perform [72].

2.1.2 Color constancy

In a previous example, we assume our scene is illuminated with equal energy pure white light so that an object's SPD includes wavelengths that are reflected. However, this is not

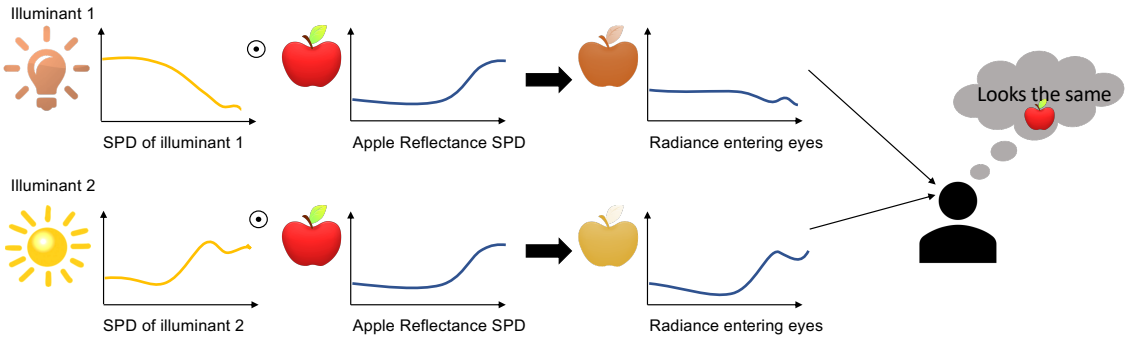


Figure 2.6: An example of color constancy of human vision. A red apple with known SPD under white light are put under two different illuminations, resulting in two different SPDs coming to the human eyes. However, human eyes can still perceive the color of the apple as it is under the white light.

the case in a real scene where we find that there are different kinds of illuminants rather than white light, affecting the perceived color of the object. In such a case, the object's SPD is comprised of its reflectance properties and scene illumination. The human visual system has an innate mechanism known as "color constancy" that allows us to compensate for the scene illumination. Color constancy allows us to perceive scene colors the same under different illuminations [13]. For example, the SPD of the reflectance of radiant energy from an apple illuminated by the sun vs. a tungsten light bulb will be significantly different, but our visual system still can perceive the object, the apple, as the same color (see Figure 2.6). Similarly, a 'white" paper would appear white even when observed in sunlight or tungsten light. In the case of white paper, the viewer is essentially directly observing the scene illumination.

When reproducing color on a display device, it is necessary to compensate for the illumination in the viewing environment to ensure accurate color reproduction, given the observer's color constancy mechanism. In particular, color spaces are encoded with an

assumed "white point". The CIE has established several SPDs to represent real illuminants that are used to establish white points. For example, CIE illuminant A is for tungsten-filament lightning; illuminant B is for noon sun light; illuminant C is for average daylight; illuminant D series represent natural daylight at various color temperatures (D50 stands for 5000K, D55 stands for 5500K, and D65 stands for 6500K). sRGB color space is defined with a white point of D65. This means that it is assumed that the viewer is observing the displayed sRGB image with an illumination similar to D65. Thus, the sRGB value R=G=B=1 (i.e., white) should be rendered as D65 sunlight. In turn, if the viewer observes the scene under D65, their color constancy mechanism would perceive anything displayed as D65 as white (i.e., sRGB R=G=B=1 is perceived as white). Therefore, it is important to be mindful of the different white points when mapping between color spaces.

2.1.3 RGB color spaces and their color gamuts

sRGB

	Red	Green	Blue
X	0.436	0.385	0.143
Y	0.222	0.717	0.061
Z	0.014	0.097	0.714

Adobe RGB

	Red	Green	Blue
X	0.610	0.205	0.149
Y	0.311	0.626	0.063
Z	0.019	0.061	0.745

ProPhoto RGB

	Red	Green	Blue
X	0.798	0.135	0.031
Y	0.288	0.712	0.000
Z	0.000	0.000	0.825

RGB color spaces and their three primary color values in CIE XYZ

Figure 2.7: The differences of three primary color values of various RGB color spaces in CIE XYZ.

Although CIE XYZ is a canonical color space representing all visible colors, practical color encoding rarely works directly with XYZ. Instead, color spaces based on three primaries (typically denoted as RGB) dominant imaging and display-related industries. CIE XYZ is still useful, as it gives us a mechanism to express the R, G, B primaries that make up a particular color space. As previously mentioned, the span of visible colors produced by

color spaces RGB primaries is called a color gamut. Note that the RGB values do not tell us about the quantitative meaning of the color, but CIE XYZ values do (see Figure 2.7).

There are several well-known RGB color spaces. We have already mentioned the standard RGB (sRGB), which has been most widely used on monitors and the World Wide Web till today. As stated earlier, the sRGB color space can encode only around ∼35% of the human gamut. Adobe introduced a wider-gamut color space, referred to as AdobeRGB [2]. The AdobeRGB color space could encode ∼50% of visible colors, which is considered medium-gamut color space. Another medium-gamut color space is Display P3, which was introduced by Apple based on DCI-P3 color space [87] and is used for most of Apple products' displays. There is also a wide-gamut color space, ProPhoto RGB [102], which is developed by Kodak. This wide-gamut color space can encompass over ∼90% of all visible colors. ProPhoto RGB is a large-gamut device-independent color space, which is linked to the ICC profile connection space. In some cases, digital cameras use ProPhoto RGB instead of CIE XYZ for PCS. Figure 2.8 compared popular RGB color spaces' gamut in CIE xy chromaticity diagram.

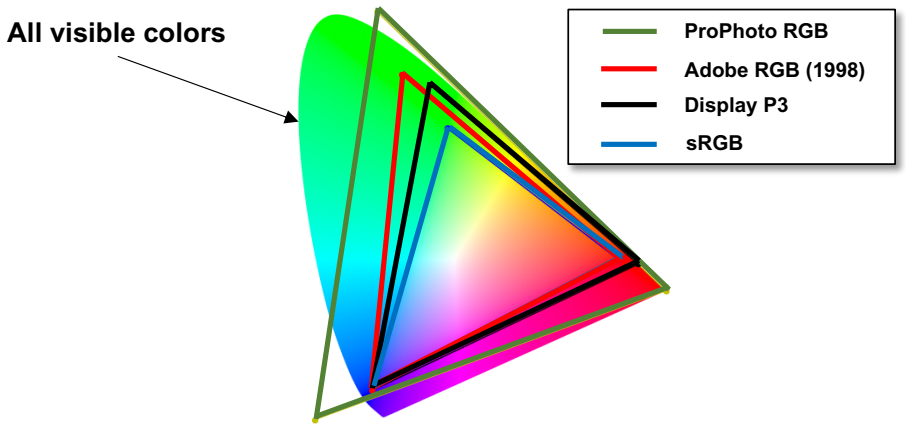


Figure 2.8: CIE-xy Chromaticity Diagram shows all visible colors and color spaces' gamut, such as ProPhoto RGB, Adobe RGB, Display P3, and sRGB.

RGB color spaces' gamuts can be visualized as 2D regions in CIE xy chromaticity diagram. However, they can also be illustrated as 3D shapes in CIE $L^*a^*b^*$ space, giving us a better view of gamuts. Figure 2.9 visualizes the ProPhoto RGB and sRGB gamuts in 3D CIE $L^*a^*b^*$, and its corresponding 2D visualization in CIE xy chromaticity diagram.

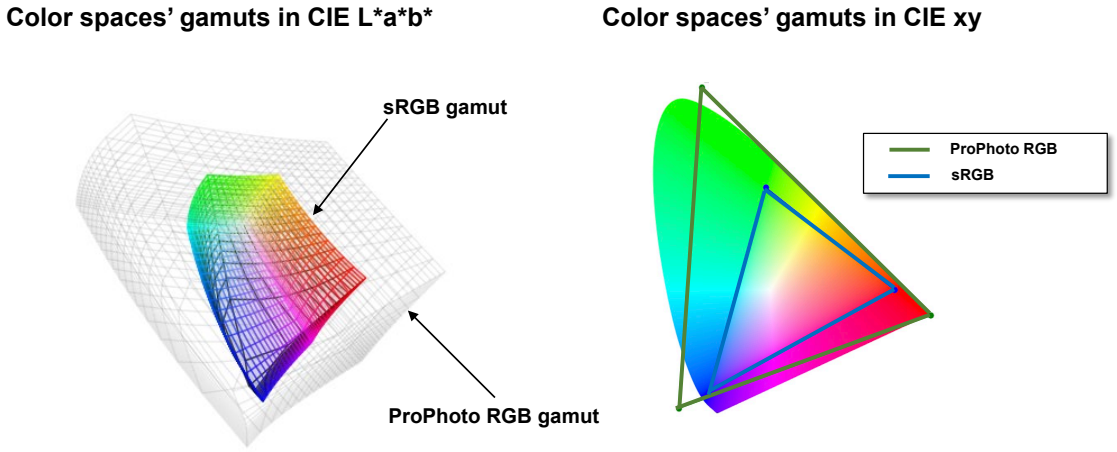


Figure 2.9: Visualization of ProPhoto RGB's and sRGB's gamuts in CIE Lab and CIE xy.

2.1.4 Gamut mapping

Gamut mapping algorithms are used to map between different sized gamuts [73]. From Section 2.1.3, we know that different RGB color spaces have different sized gamuts. The mismatch in gamut sizes poses a challenge for color reproduction. For example, suppose we want to reproduce a color image viewed in a device with medium-gamut color space in another device capable of encoding images to a much smaller gamut color space. In that case, we need an algorithm to ensure that colors from the source device are adapted to fit inside the available color gamut under destination conditions. The next chapter will provide more details on gamut mapping.

2.2 Camera Processing Pipeline

Cameras must process their sensor response to scene radiance to a 2D pixel grid encoded in some standard color space for display and exchange. As a result, it is necessary to describe the set of processing steps applied onboard cameras, collectively referred to as the camera processing pipeline. The work in this dissertation will exploit the availability of a wide-gamut image representation used by cameras when processing the sensor image.

A camera uses dedicated hardware, called an image signal processor (ISP), to perform a sequence of steps to transform raw sensor data captured with color filter arrays to the final display-referred output [4, 31, 59]. Figure 2.10 provides an overview of a typical ISP. First, the ISP applies a linear transformation to a raw-RGB image in the Bayer processing stage: pre-processing (black level offset, normalization, lens shading), white balance, demosaic, noise reduction, and sharpening. The image values are still linked tightly to the recorded scene data in this raw-RGB state. Then, ISP transforms the raw-RGB color values to a common perceptual color space, CIE 1931 XYZ or the wide-gamut ProPhoto RGB. The next stage is photo finishing or enhancement in which ISP performs non-linear transformations to the image, namely color manipulation, tone mapping, final color space conversion (e.g., Adobe RGB or sRGB), and gamma curve application. The image's final output color space usually has a much smaller gamut than that of its internal state. Figure 2.10 shows typical steps that ISP performs on an image.

As described above, most commercial cameras encode the final output color space as sRGB. Showing such an image in a larger-gamut device would raise a gamut-mismatch issue. That is the reason why we take advantage of the insight of ISP and use software that emulates ISP to reach the wide-gamut representations of images to generate ground-truth targets for our wide-gamut restoration algorithms.

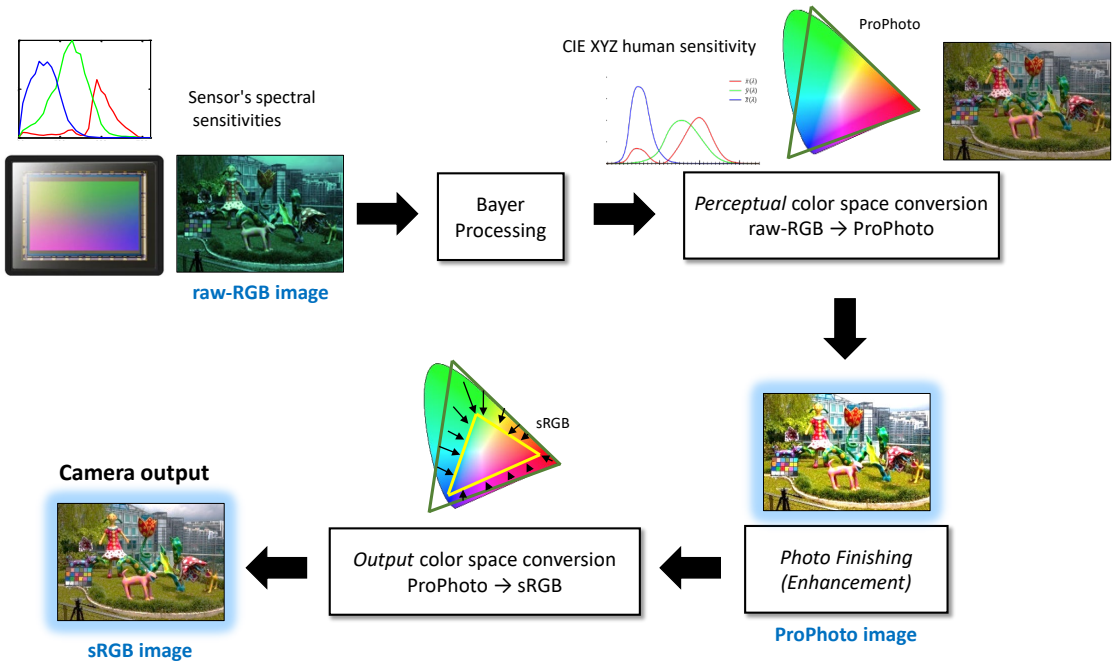


Figure 2.10: This figure illustrates a typical imaging pipeline implemented by a camera's image signal processor (ISP) hardware. An essential step in the pipeline is converting the sensor-specific raw-RGB colors to a perceptual color space based on CIE XYZ (i.e., ProPhoto). The ISP then enhances the wide-gamut ProPhoto-encoded image. The enhanced ProPhoto image is finally converted to its output color space. While some cameras allow an option to save the image in a medium-gamut color space, such as Adobe RGB or Display P3, most cameras default to sRGB. By emulating this camera pipeline, we can obtain training data from pairs of the same images encoded in small-gamut sRGB and wide-gamut ProPhoto (highlighted in blue).

3 Related Works

3.1 Introduction

In this chapter, we provide an overview of gamut mapping algorithms. We focus on two main types of gamut mapping based on the relationship between source and destination gamuts: (1) gamut reduction and (2) gamut expansion. As we will see, there has been significantly less work targeting gamut expansion. Furthermore, we highlight some studies that are closely related to our task and those that have influenced our approaches.

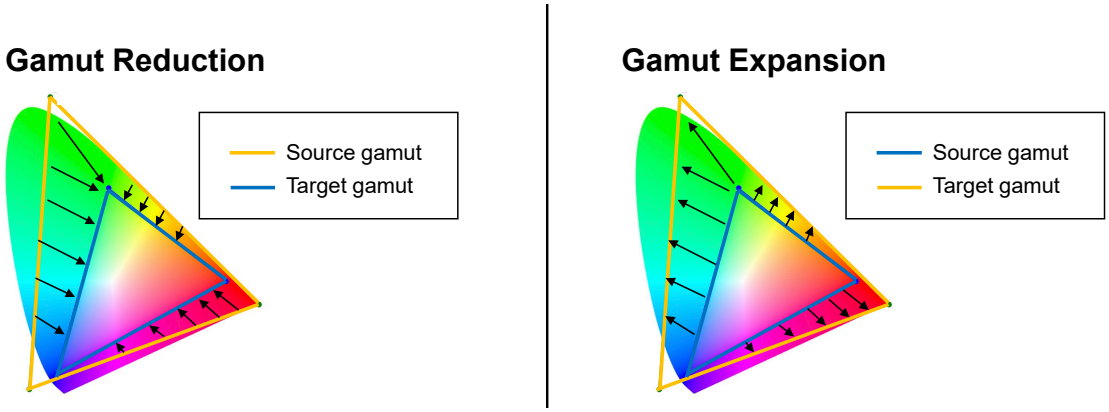


Figure 3.1: Comparison between gamut reduction and gamut expansion.

3.2 Gamut Reduction

Gamut reduction is applied when the source gamut is larger than the destination and aims to map the source colors to the destination region (see Figure 3.1). This is the most well-studied type of gamut mapping. Gamut reduction could be classified into two main categories: color-by-color reduction and spatial-aware reduction [72]. The former deals with each pixel separately, while the latter considers the spatial relationship between pixels in the input images.

3.2.1 Color-by-color reduction

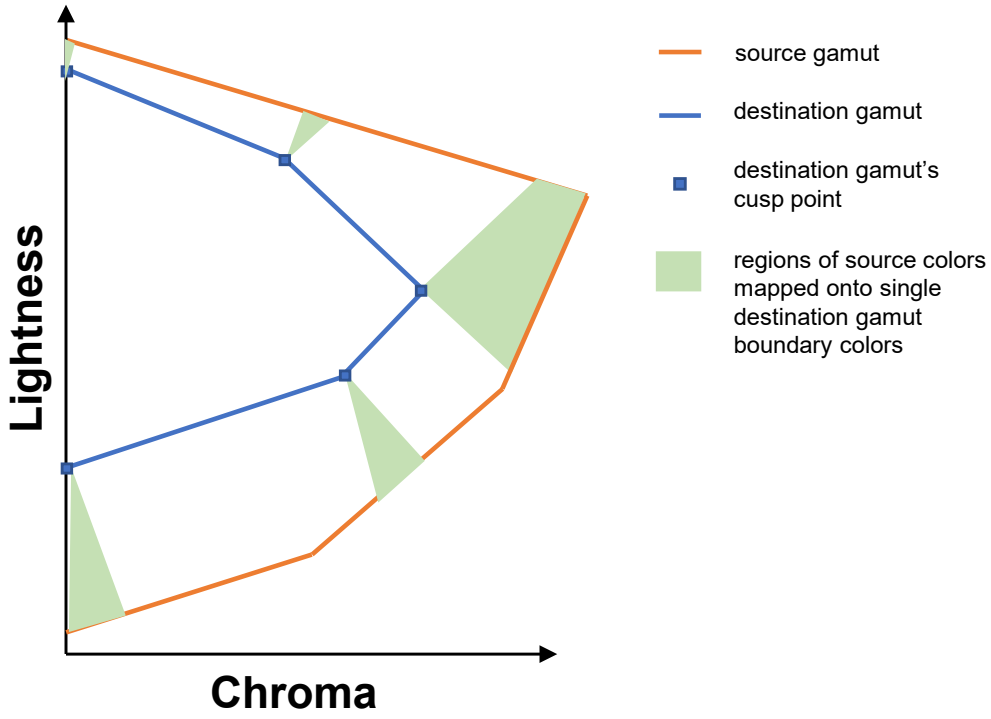


Figure 3.2: Overview of the minimum color difference gamut clipping method performed in planes of constant hue of CIE LCh color space.

The color-by-color gamut reduction algorithms could be divided into two subtypes: clipping and compression. Clipping is the simplest way for gamut reduction as it tries to minimize colorimetric errors by projecting or clipping colors in a wider gamut that are outside of a smaller target gamut to the boundary of the smaller gamut. Figure 3.2 visualizes how the minimum color difference gamut clipping is done. Source colors out of the destination gamut are mapped to their counterparts in the destination gamut's boundary to ensure their Euclidean distance between them is the smallest. Some regions of source colors are clipped onto single cusp points in the destination gamut's boundary. Source colors that lie inside the destination gamut are kept unchanged. By doing these steps, the source image is mapped to a destination gamut in which each output pixel's color is as close to its original value as the destination gamut allows.

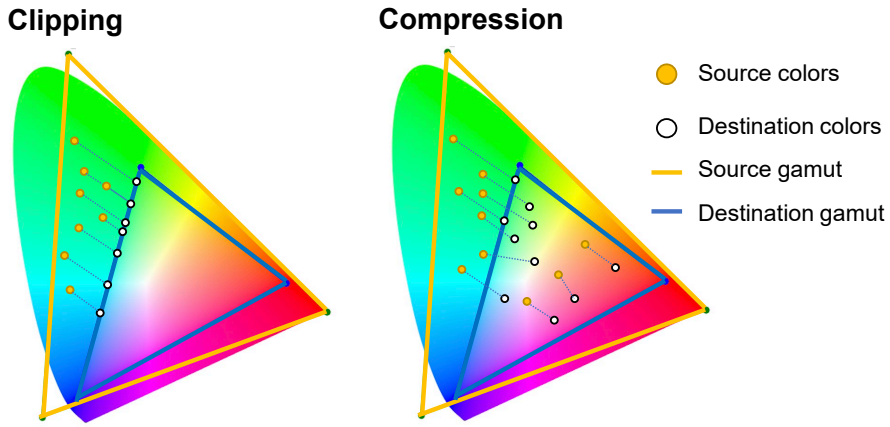


Figure 3.3: Comparison between clipping and compression algorithms for gamut mapping.

Algorithms based on gamut compression are popular in the gamut-mapping literature. These reduction algorithms not only modify the out-of-destination-gamut colors but also change colors inside the destination gamut. Figure 3.3 illustrates the general difference between the clipping and compression algorithms.

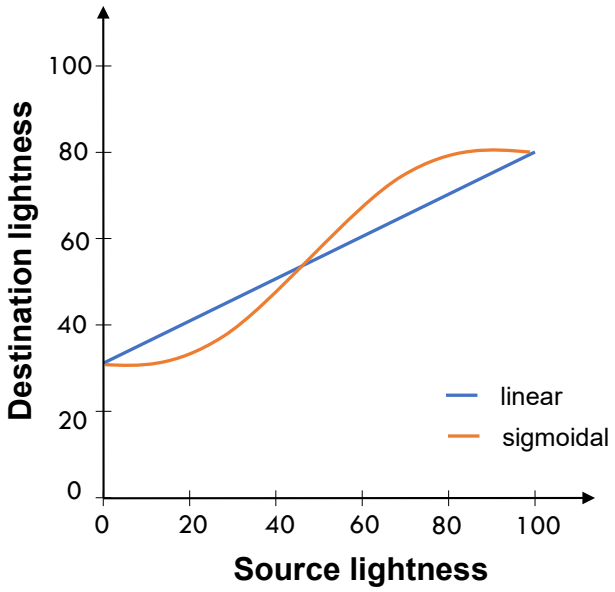


Figure 3.4: Compare between linear and sigmoidal lightness mapping from source lightness to destination lightness.

For color-by-color gamut compression algorithms, the mapping is performed separately for each dimension of the mapping color space, usually the CIE $L^*C^*h^o$ (lightness, chroma, and hue angle). Some works propose sequential mappings that reduce lightness first, followed by chroma mapping, with the hue unchanged. The lightness dimension gets modified by applying a mapping, calculated by the lightness range of the source and target gamuts, to all input image's lightness values. Some approaches use a simple linear transformation for this lightness mapping [45, 88, 97], but it leads to a loss in image contrast [72]. Others use non-linear functions such as sigmoid [15] to deal with such a contrast loss (see Figure 3.4). Braun et al. [15] proposed a sigmoidal mapping function that is image-dependent. In particular, they carried out a psychological experiment to estimate mapping function parameters that were adjusted by participants so that a final output image in a smaller gamut color space has the lightness channel as close to the original image as possible.

Braun et al. found a correlation between the mapping values selected by participants and image statistics, namely the lightness corresponding to the 75% point of the source image's cumulative lightness histogram and the black destination points, which help to determine parameter values for their function.

Another strategy for lightness mapping is to make the mapping depend on the source image's chroma. For example, in the previous lightness mapping used by [15], the lightness mapping is applied equally across the whole color space. Morovic [70,71] then proposes the chroma-dependent lightness mapping to address this issue, in which a target lightness is computed by the weighted sum of a given source lightness and its fully lightness compressed version using either linear or sigmoidal. The weighting factor is chroma dependent, which takes the maximum value of 1 at the neural axis of the color wheel of CIE $L^*a^*b^*$ space and then falls out as chroma increases (see Figure 2.5).

After modifying the lightness channel, the next step is to transform the chroma values. There are various ways to address this task, such as scaling all chroma values with a single factor [93], scaling the chroma with a hue-angle-dependent function [1, 44], using a linear function on two or three different segments, or applying nonlinear functions [34, 45, 106]. The nonlinear functions approach is quite effective as it can modify the out-of-gamut points while leaving the in-gamut points unchanged.

For example, Johnson [43] proposed a typical example of a gamut reduction algorithm, called LLIN which is short for lightness linear, that first linearly scales the lightness followed by linearly scaling the chroma in a plane of constant hue. Figure 3.5 illustrates steps of the LLIN method, which first maps the source's lightness to the range of the destination's lightness, then the intermediate colors are compressed further along the lines of constant lightness and hue to ensure that the maximum source chroma is similar to the maximum destination chroma. Compared with the simple gamut-clipped procedure, LLIN

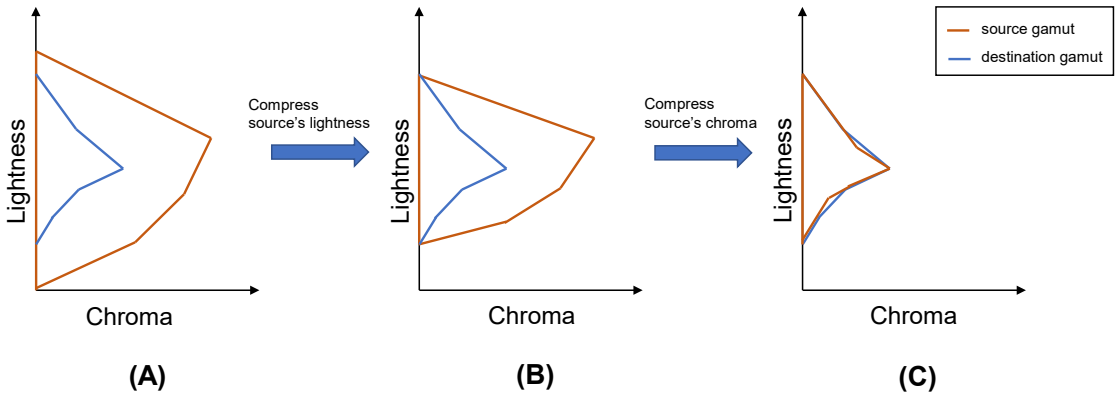


Figure 3.5: Overview of Johnson's LLIN method [43] for gamut reduction (adapted from [72]). (A) The source and destination gamuts are shown in terms of lightness and chroma in a plane of constant hue. (B) The source gamut's lightness is compressed to the range of the destination gamut's lightness. (C) The lightness-compressed source gamut's chroma is compressed to fit inside the destination gamut boundary.

can preserve more detail and relationships within an image, making the input image have less contrast and appear less colorful.

Instead of mapping the lightness channel first followed by mapping the chroma, another way is initial chroma mapping following the lightness mapping [70]. While the hue angle is usually kept unchanged in the majority of works, Johnson [46] proposed a method – termed CARISMA – which suggests changing the hue can lead to the greater similarity between the input and the reproduced colors by observing professionals reproducing colors manually. That finding is later used in many other algorithms [74, 98].

Instead of the sequential modification described above, some works apply simultaneous mapping of lightness and chroma in constant-hue planes. Firstly, the most basic simultaneous gamut reduction algorithms compress source values toward a single point, which is a so-called focal point. The simplest implementation is choosing the focal point at the center

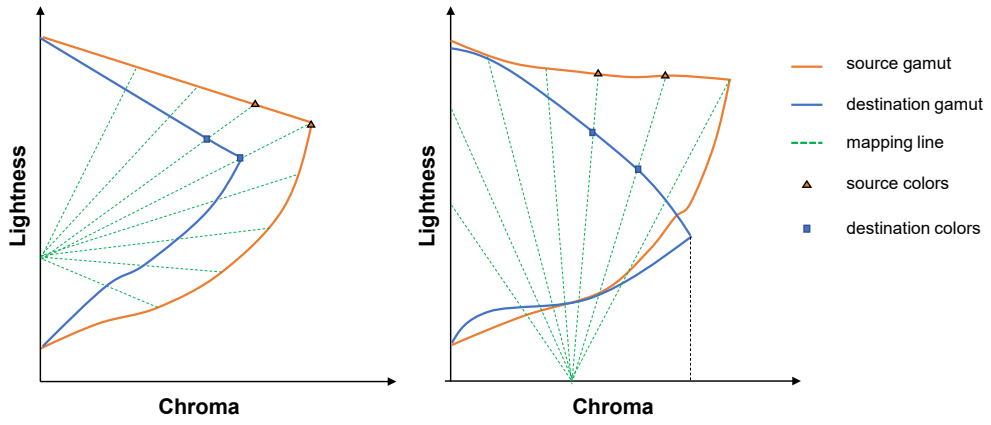


Figure 3.6: Visualization of a gamut compression algorithm that map source values toward focal points [46] (adapted from [72]).

of the lightness axis [45, 64, 65]. Other implementations choose different focal points. For example, CARISMA [46] algorithm chooses two different focal points; the first is on the lightness axis where the line connecting the input and target gamuts' cusps intersects it, and the second is on the chroma axis that has half the chroma of the destination's cusp at the given hue (see Figure 3.6). In addition, many gamut reduction algorithms split source colors into many smaller regions and use different transformations to each region at a given hue plane. More specifically, some works define a "core" region within the overlap between the source and target gamuts in which nothing is compressed, and other regions with different single focal points toward which colors are compressed [45, 49, 81]. Figure 3.7 illustrates compression algorithms with a core region. Keeping the core regions unchanged gives good results in the end as it combines benefits of the clipping and full compression algorithms as well as can represent some important memory colors (skin tones and neutral colors) if the core region is set properly [72].

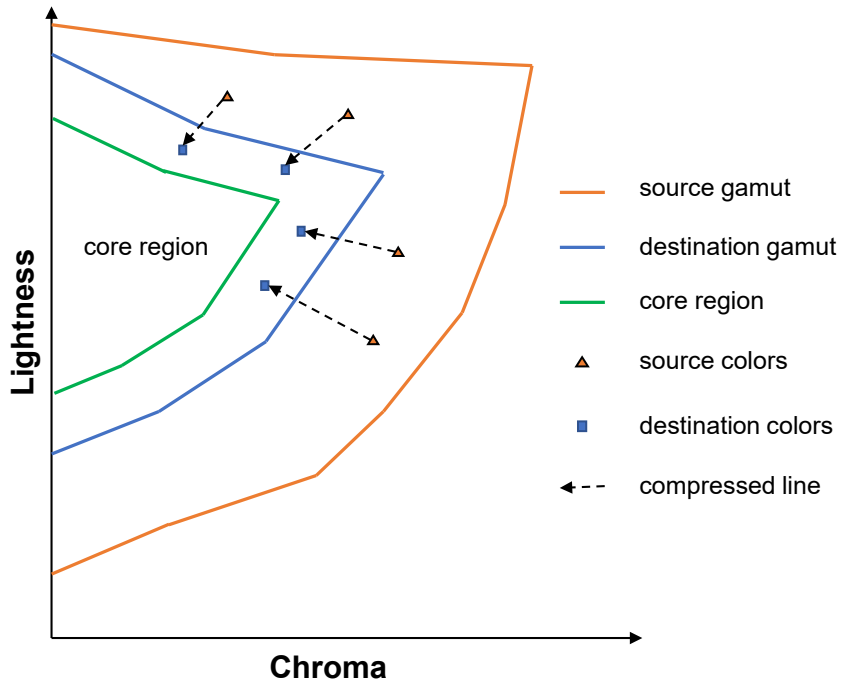


Figure 3.7: Visualization of a core region in which nothing is compressed in some gamut reduction algorithms (adapted from [72]).

3.2.2 Spatial-aware gamut reduction

The previous section described gamut clipping and compression algorithms that did not explicitly consider spatial image content. These methods compress a whole range of input colors to a single target color in many cases. As a result, some of the image detail in a many-to-one region is lost in the output image. To preserve local detail, gamut mapping algorithms have been proposed that consider the local neighborhood relationships among image pixels [72].

To this end, several methods follow what is termed "frequency-sequential," which applies color-by-color gamut reduction algorithms on a low-pass filtered input image and then adds in the high-pass filtered input image to form a final output [9, 10, 67, 75]. By using high-

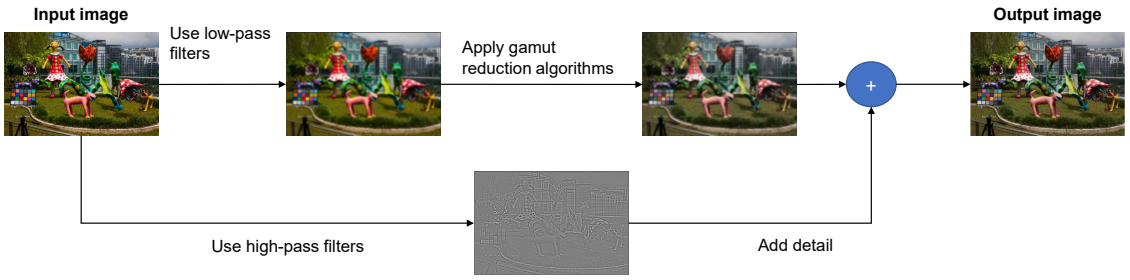


Figure 3.8: Overview of spatial-aware gamut reduction.

pass filters, these methods directly map image details obtained from the high-pass filtered to the gamut reduced output image. Figure 3.8 illustrates the typical pipeline of these approaches.

Other strategies involve methods that consider gamut mapping as an optimization problem [77, 78]. These approaches use band-pass filters for the Fourier-transformed CIELAB input and target images, then use various metric functions that consider the difference between the two images by weighting differences in the various bands according to the human contrast sensitivity lightness, yellowness-blueness, and redness-greenness. These spatial gamut reduction algorithms have the potential to deliver better reproductions than color-by-color algorithms, especially in the cases that require subjective accuracy [72]. The drawback of these methods is that they require more computational time than methods that only process the image colors.

3.3 Gamut Expansion

There is significantly less work targeting gamut expansion as compared with gamut reduction. However, given the recent technology advances in display technology, there has recently been a renewed interest in gamut expansion methods. For example, gamut expansion is wildly applied to imagery being displayed on wide-gamut displays such as ultra-

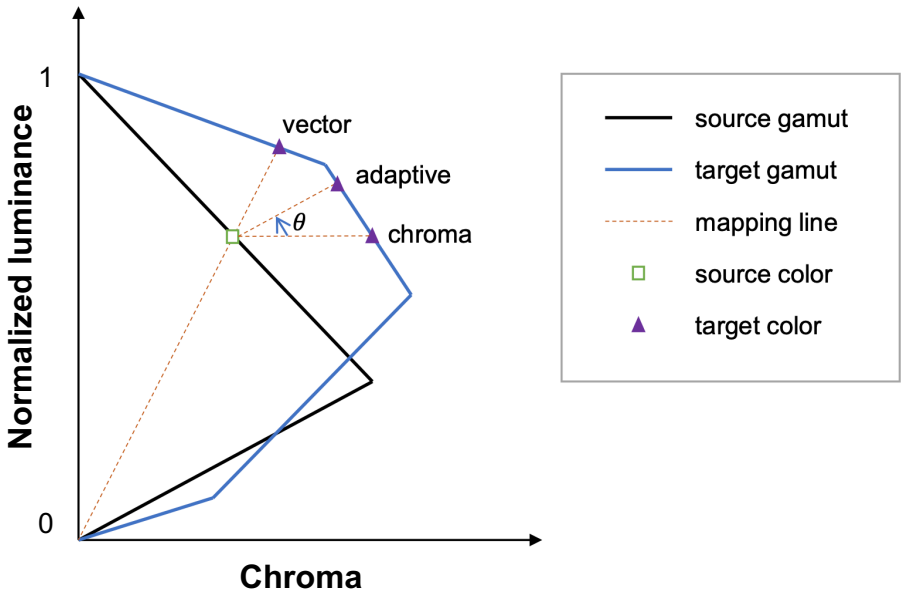


Figure 3.9: The figure illustrates three strategies of gamut expansion from Kim et al. [50]: vector mapping, chroma mapping, and adaptive mapping (reproduced from [72]).

high-definition (UHD) TVs. Gamut expansion is usually cast as an image enhancement problem and not a gamut restoration problem. This is because we typically have no information regarding an image's original wide-gamut representation. As a result, gamut expansion attempts to modify the image to produce visually pleasing image in the wide gamut as shown in Figure 3.1.

One simple approach for gamut expansion is to apply any gamut reduction in a reverse direction [14, 35], however this way may introduce unnatural artifacts and unpleasant experience. There are some gamut expansion algorithms propose specifically for adapting small-gamut printed images to the wide gamut HDTV [37, 38]. They first map the lightness using non-linear tone curve, and extend the chroma along lines of constant hue and lightness. Some approaches take advantage of observer experimental data to determine

functions for gamut extension [7, 47], while others try to classify each image's content into different color regions and perform gamut expansion accordingly. For example, Pan et al. [89] label each color region in an image as non-skin or skin, Casella et al. [17] separate low and high chroma objects from each other, and Heckaman et al. [32] identify some memory colors in an image namely blue sky or green grass and treat them differently. Kim et al. [50] propose three ways of gamut extensions for HDTV, which apply the mappings globally to images on the luminance-linear space. Their three ways of mappings are: vector mapping (extension along lines from the origin through source colors), chroma mapping (along lines of constant illuminance and hue angle), and adaptive mapping (along lines at a specific angle θ between the first two strategies). Figure 3.9 visualizes their three ways of gamut extensions.

Apart from global algorithms above, there are other works that extend colors by considering the spatial distribution in given input images, resulting in more adaptive and flexible gamut extension algorithms at the cost of computational resources and processing time. Yihui Li et al. [58] propose two stages gamut extension for wide-gamut displays. The first stage is applying a global mapping with hue-varying non-linear extension function, and the second stage is image-dependent chrominance smoothing which detect and revise only the over-contrast areas in images by employing a varying high-pass filter. Zamir et al.(2016) [113] conduct spatial gamut extension by applying a monotonically increasing function on the saturation channel (input images are converted to HSV color space), which can extend normally high-saturated-color objects and keep low-saturated colors, such as skin tones, neutral colors and some special memory colors, almost unchanged. Zamir et al.(2017) [114] also propose an iterative gamut extension algorithm for cinema which is only applied on the chromatic components of CIE $L^*a^*b^*$ color space and check constraints per pixel after each iteration to prevent distortions in hue, saturation, and chroma.

Recently, Zamir et al. (2019) [115] proposed a framework for gamut mapping based on finding in human vision. Specifically, their method first processes the saturation channel by convolving it with a center-surround linear filter. Next, a rectification is applied to ensure the saturation does not decrease. Then, the brightness is modified to compensate for the Helmholtz-Kohlrausch effect which is the phenomenon in which two colors have the same luminance and hue but different chroma appear to have different perceived brightness.

To the best of our knowledge, no work in the literature tries to restore the reduced wide-gamut colors to their original form. Most of the gamut extension approaches aim to enhance the colors as perceptually pleasing as possible. As a result, existing methods produce results that are far from close to the original colors. For example, Figure 3.10 visualizes the difference between the ground-truth ProPhoto RGB colors and their restored values after being clipped in the CIE Lab color space using Zamir et al.'s (2019) method [115]. It is easy to notice that the restored values try to expand across the ProPhoto gamut as much as possible. However, the expanded colors are not close to their ground truth.

3.4 RAW Recovery/Derendering

Also related to our task are approaches for sRGB *derendering*, where the goal is to recover the original RAW sensor image from an sRGB input. Early approaches to this problem carefully modeled the in-camera rendering process to perform derendering [18, 19, 29, 51]. These methods have a drawback that they demand extensive calibration routines, which need to be conducted per camera and per camera setting. Recent deep learning approaches to raw reconstruction, such as [63, 79], also face similar challenges. The models trained are specific to each camera, requiring a substantial amount of training data for every individual camera.

Several recent studies [5, 80, 82, 83, 92, 112] have recommended embedding additional

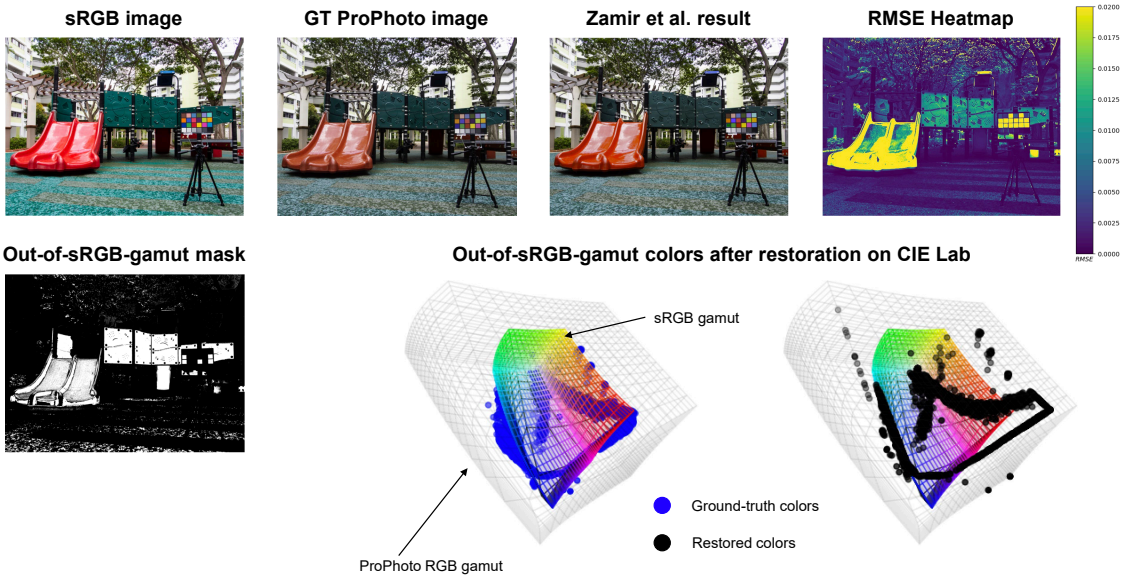


Figure 3.10: Comparison between ground-truth ProPhoto image and Zamir et al.'s restored output. Out-of-sRGB-gamut colors after restoration using Zamir et al.'s method [115] and their corresponding ground truth are shown in CIE Lab color space. The heatmap of root mean square error (RMSE) between ground-truth image and Zamir et al.'s output are also shown.

metadata into the output sRGB image at the time of capture. This extra data enables more precise raw reconstruction compared to the generic methods previously mentioned. Yuan and Sun [112] suggest saving a small raw image at either a quarter or half the size of the full-resolution sRGB image and then conducting a guided upsampling from this lower-resolution raw image to reconstruct the full-resolution raw version. Nonetheless, the size of the small raw file, ranging from 1.5 to 6 MB, restricts the practical applicability of this technique.

Nguyen and Brown [82, 83] calculate and save a collection of raw reconstruction parameters during capture, derived from the corresponding raw-sRGB image pair. These parameters replicate common ISP operations like white balance adjustment, color space

conversion, global tone mapping, and gamut mapping. However, their optimization algorithm, when applied to full-resolution images, incurs significant computational expense and poses challenges for real-time on-device implementation.

Afifi et al. [5] introduced a multi-pass imaging framework in which a downsampled, demosaiced raw-RGB image undergoes processing by the ISP multiple times under varying settings. Subsequently, mapping functions are determined between each distinct rendering and a downsampled version of the final sRGB image. The researchers show that storing these mapping function parameters as metadata at the time of capture facilitates precise raw recovery or modifications to the image after capture. Nevertheless, the limitation lies in the fact that most ISPs are not designed to handle multiple passes, and implementing their suggested framework would require significant alterations to current designs.

Punnappurath and Brown [92] suggested employing a small, uniformly sampled set of raw values as metadata. They introduced a spatially aware recovery algorithm, enhancing the robustness of their method against local tone mapping and various non-global ISP adjustments. However, their approach to raw reconstruction relies on interpolating with a 5D radial basis function, which tends to be slow in practical applications.

Nam et al. [80] introduced a framework that closely resembles the one in [92], but with a key difference: they do not confine their sampling to a uniform grid. Their approach encompasses learning the sampling and reconstruction processes in a holistic, end-to-end fashion. Additionally, they design the reconstruction network to be adaptable, enabling further refinement using the sparse metadata samples.

Our methods are similar to works that save small amounts of specialized metadata in the sRGB image to assist in the recovery problem. They do not require significant modifications on the current ISP design and are optimized for real-time applications. Such metadata can be in the form of a parametric model [82, 83] or RAW pixel samples [80, 92].

While having access to a reconstructed RAW image would allow it to be re-rendered back to a wide-gamut ProPhoto format, such rendering requires camera-specific photo-finishing parameters that are often not readily available.

3.5 Image-to-image Translation

Image-to-image translation methods have garnered widespread popularity and attention within the fields of computer vision and machine learning, largely due to the significant advancements in Generative Adversarial Networks (GANs). Since the inception of GANs by Goodfellow et al. [30] in 2014, a myriad of variations and improvements like DCGAN by Radford et al. (2015) [94], Wasserstein GAN (WGAN) by Arjovsky et al. (2017) [8], and CycleGAN by Zhu et al. (2017) [117] have been developed, enhancing the capability, stability, and quality of image generation and translation. These advancements have made it possible to perform complex image translations, such as photo-realistic image synthesis, style transfer, and the transformation of sketches into detailed images, pushing the boundaries of how machines understand and manipulate visual information.

The exploration of image-to-image translation can be primarily divided into supervised and unsupervised directions. Supervised methods [42, 91, 99, 107], which rely on paired datasets comprising corresponding images from source and target domains, have been foundational in establishing the field. Phillip Isola et al. [42] introduced a general-purpose solution to image-to-image translation problems using conditional adversarial networks. This method, known as Pix2pix, employs paired data to learn a mapping from input to output images. It has become foundational in the field for tasks like translating sketches to images, black and white photos to color, and more. Ting-Chun Wang et al. [107] developed Pix2pixHD, an extension of Pix2pix, which is capable of generating high-resolution images from semantic label maps. Pix2PixHD enhances the original framework to support high-

definition image output, making it particularly useful for tasks requiring detailed image synthesis, like photo-realistic image generation from semantic layouts. The improvements in Pix2PixHD are underscored by the adoption of a novel adversarial loss and the implementation of new architectures for the generator and discriminator, designed to operate effectively across multiple scales. However, the challenge of obtaining paired datasets has led to a surge in unsupervised approaches [6, 39, 61, 117], where unsupervised models have made it feasible to translate images between domains without one-to-one correspondences.

Within the scope of gamut restoration, employing a supervised image-to-image translation method becomes a viable strategy to tackle the issue of gamut mismatch, particularly when paired data is available, a possibility afforded by the understanding of the camera pipeline discussed in the previous chapter.

3.6 Implicit Neural Representations

Implicit neural representations (INRs) have become a powerful method for representing complex data in a compact and computationally efficient manner recently. Different from conventional representations that explicitly store data in discrete structures like arrays or matrices, INRs take advantage of the capacity of neural networks to implicitly encode data within the parameters of the network itself. The core idea behind implicit neural representations is to use a neural network to model a continuous, differentiable function that maps coordinates (e.g., spatial positions, time) to output values (e.g., color, density, temperature) of the data being represented. While several studies have already explored using coordinates as inputs to neural networks [62, 90, 101, 108, 109], INRs have gained their greatest popularity with the introduction of methods in 2020 like SIREN (Sinusoidal Representation Networks) introduced by Sitzmann et al. [100], and NeRF (Neural Radiance Fields) by Mildenhall et al. [68]. These two methodologies have not only showcased the

potential of INRs but have also laid the groundwork for their diverse applications across various domains. In SIREN, Sitzmann et al. [100] explored the use of periodic activation functions, specifically sine functions, within neural networks to better model complex and smooth signals. SIREN was a foundational work that demonstrated the power of INRs for a variety of tasks. On the other hand, Mildenhall et al. [68] with NeRF demonstrated how INRs could be used to synthesize novel views of 3D scenes from a set of sparsely sampled images. By modeling the volumetric scene function of a space, NeRF could generate photo-realistic renderings of 3D scenes from viewpoints never captured in the original dataset. This groundbreaking approach revolutionized 3D rendering techniques, leading to many improved versions that build on its original achievements [12, 20, 24, 60, 66]. INRs have also been applied to a range of tasks including representing textures [86], the flow of 3D shapes [85], audios and differential equations [100].

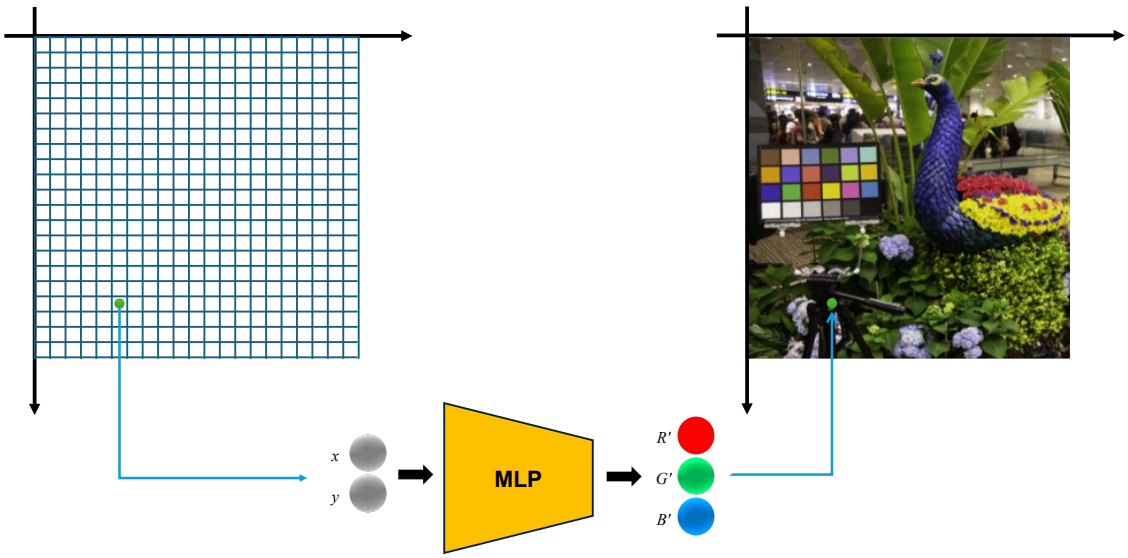


Figure 3.11: Figure shows how INR views a 2D image as a continuous, differentiable function, taking coordinates as inputs and outputting color values through a MLP.

In the context of 2D images, INRs treat an image as a continuous function, which can be approximated by a neural network [100, 105], rather than as arrays of pixels in two dimensions. This approach offers a genuine continuous representation of the original 2D signal, as opposed to the discretized and quantized version produced by traditional pixel-based representations. Figure 3.11 demonstrates how an INR approaches a 2D image by treating it as a continuous, differentiable function that accepts coordinates as input and produces color values as output, using a MLP.

While training an MLP to serve as an INR for a scene or an image is computationally cheap, it takes a considerable amount of time for the model to converge (training a model can take several hours, and generating a single image may take as long as 30 seconds [68]). Thus, a range of studies focused on accelerating the optimization process for MLPs, enabling the use of INRs in real-time application, particularly in real-time view synthesis. Many studies attained real-time rendering by precomputing view-dependent colors and opacities into sparse volumetric data structures [28, 33, 111]. Other studies in real-time view synthesis demonstrated that both rendering and training speeds can be greatly increased through various parameterization methods of a radiance field [25,27,103,110]. Most interestingly, the works of Müller et al. [25, 76] proposed methods for efficient input data encoding, making it possible to use a much smaller MLP than NeRF. Techniques like meta-learning were also adapted to pre-train MLPs on a range of tasks or data distributions, allowing for faster adaptation to specific INRs with fewer gradient updates [57, 84].

3.7 Summary

We carefully looked at studies related to our topic, focusing on the detailed aspects of gamut reduction and gamut expansion. We observed a lack of extensive research dedicated to gamut expansion. We also covered tasks that are closely connected to our work and re-

viewed several studies that have shaped our methods, namely RAW recovery/derendering, image-to-image translation, and implicit neural representation. In the upcoming chapters, we will present three methods aimed at achieving gamut expansion through restoration.

4 Gamut Restoration via Deep Learning

In this chapter, we outline our first approach, which employs a deep-learning framework for color gamut restoration. We begin with an introduction. This is followed by an overview of our methodology, and then we delve into the specifics of the network architecture underpinning our approach. In the section dedicated to experiments, we detail the dataset used and the results obtained. Our findings demonstrate that this deep-learning strategy yields encouraging outcomes. The source code and dataset of this work can be found on Github: `https://github.com/gamut-mapping/GamutNet`

4.1 Introduction

A gamut conversion is required to map between color spaces. Converting from wide-gamut to small-gamut color spaces requires deciding how to contract and clip color values to lie within a smaller gamut boundary. Conversely, the restoration of a wide-gamut color space from a small-gamut one is more challenging as the correct target colors in the wider gamut are unknown (see Figure 4.1). Thus, when converting an sRGB image to a wider-gamut color space, most software and devices employ a *colorimetric* strategy that strives for accurate color reproduction in the target color space. This conservative approach avoids color distortion, however; it leaves large regions in the wider-gamut unused. Non-colorimetric gamut-expansion methods—also referred to as saturation or perceptual intent

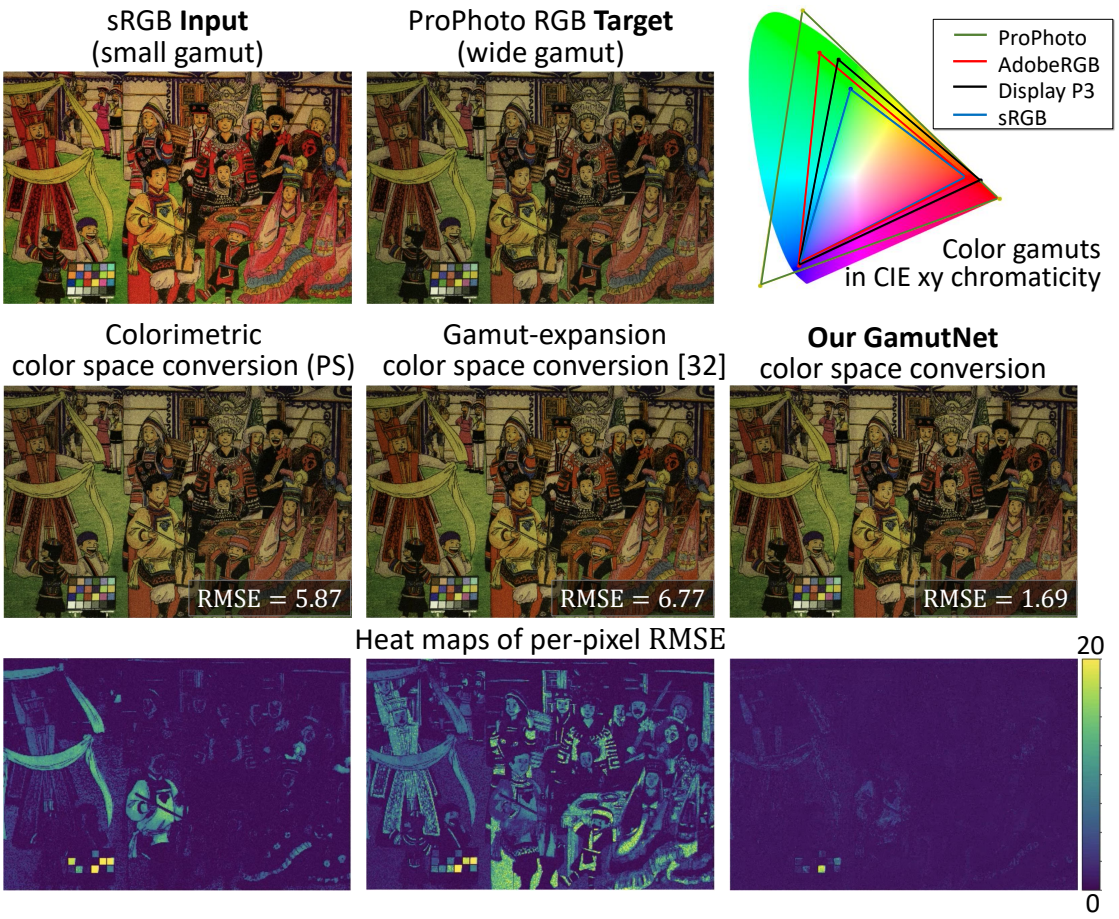


Figure 4.1: (top) An input image encoded in its small-gamut sRGB color space and its ground-truth wide-gamut ProPhoto. Gamuts of common color spaces used on cameras (sRGB, Adobe RGB, Display P3, and ProPhoto) are shown. (middle) Conversion of the sRGB image to ProPhoto using a standard "colorimetric" conversion from Photoshop [40], a state-of-the-art gamut expansion method [115], and our GamutNet result. (last) Heat maps of per-pixel errors in terms of root-mean-squared-error (RMSE).

methods—such as [53,54,115,116], are designed to stretch the input gamut colors to fit the target gamut with the goal of enhancing the images and making them appear more vivid. However, in terms of restoration, their results may introduce unwanted color distortions

as they do not take into account the ground-truth colors in the wide gamut.

The fundamental problem is that given an sRGB encoded image, there is no clear notion of what the wider-gamut colors should be. Our key insight into this problem is that virtually all cameras' image signal processor (ISP) units produce an internal version of a captured image in the wide-gamut ProPhoto RGB color space [48]. This in-camera conversion of the raw-RGB sensor image to the ProPhoto RGB color space allows cameras to perform photo enhancement in a color-rich representation. The conversion to a gamut-limited output color space is applied as the last step of the camera pipeline. Unfortunately, cameras do not provide easy access to this wide-gamut image state. However, it is possible to use software ISPs to obtain these wide-gamut ground-truth representations of the sRGB images. The access to the wide-gamut images enables the color space conversion problem to be cast as a restoration problem where known ground truth is available to learn recovery models.

We describe an approach to leverage the in-camera processing pipeline to produce a dataset of 5,000 image pairs encoded in both the small-gamut sRGB and wide-gamut ProPhoto RGB color spaces. Using this dataset, we train a DNN model, termed GamutNet, to restore sRGB image colors back to their wide-gamut ProPhoto RGB representation. We show that this deep-learning approach to color restoration can reduce errors by almost 50% over existing methods. Moreover, we show that we get improved color management for free by targeting the wide-gamut ProPhoto color space. That is, our recovered ProPhoto image can be subsequently converted to medium gamut color spaces such as Adobe RGB and Display P3, again with an error reduction of 50% compared to existing methods. Our dataset, code, and trained model will be made publicly available.

4.2 Method Overview

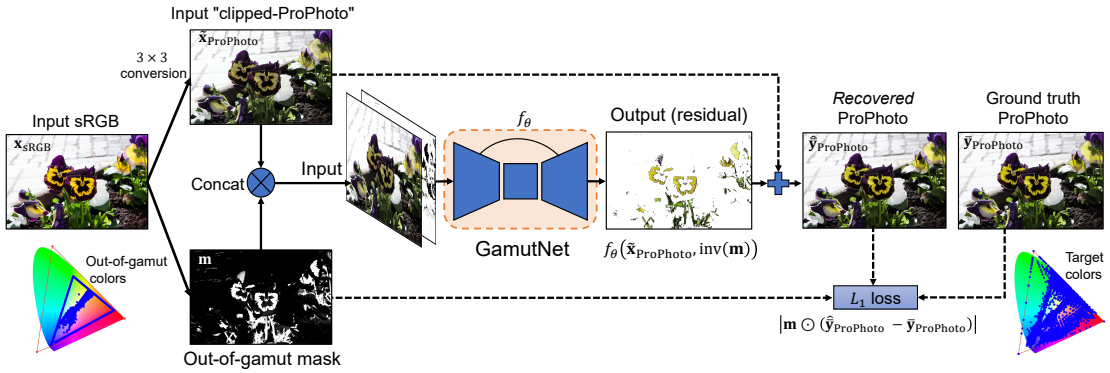


Figure 4.2: An overview of deep-gamut-restoration method. We first convert the input sRGB image into a linear "clipped"-ProPhoto image using a 3×3 linear transformation; we then compute the out-of-gamut mask; the "clipped"-ProPhoto image and mask are concatenated as input to GamutNet; GamutNet predicts the residual pixel corrections and adds them to the input clipped ProPhoto image. Our L_1 loss is computed between our recovered ProPhoto output and its corresponding ground truth on the out-of-gamut pixels only. Our GamutNet architecture is shown in the bottom part. The out-of-gamut colors are shown in CIE-xy chromaticity plots for the input and target. Note that the chromaticity plot is a projection of the 3D color values, and therefore some out-of-gamut pixels will fall inside the 2D-triangle of the gamut's projection.

Our approach works as follows. Given an image encoded in the sRGB color space, denoted as $\mathbf{x}_{\text{sRGB}} \in \mathbb{R}^{M \times 3}$, where M is the number of pixels, we want to map it into a wide-gamut image, denoted as $\mathbf{y}_{\text{PP}} \in \mathbb{R}^{M \times 3}$, in the ProPhoto color space. Note that our goal is to produce an image that is as colorimetrically accurate as possible in the wide-gamut color space. The colors that are most problematic are those that were out-of-gamut

in the in-camera conversion from ProPhoto to sRGB. The color values of these pixels had to be *clipped* to the sRGB gamut boundary. As a result, we focus our network's capacity on these pixels and not the pixels with in-gamut colors.

Linear color space transformation We first apply a pre-defined 3×3 linear transform to convert $\mathbf{x}_{\text{sRGB}}$ to a clipped version of ProPhoto. This is written as follows:

$$\tilde{\mathbf{x}}_{\text{PP}} = C\ \bar{\mathbf{x}}_{\text{sRGB}}\ , \tag{4.1}$$

where $\bar{\mathbf{x}}_{\text{sRGB}}$ represents the linearized sRGB values after a de-gamma operation is performed on $\mathbf{x}_{\text{sRGB}}$. $C \in \mathbb{R}^{3\times3}$ is a pre-defined matrix that could be decomposed into three parts: the transformation of sRGB color to the connecting space C_{sRGB}^{-1}, the chromatic adaptation matrix C_{CAT02} [69], and the transformation of the color in XYZ space to ProPhoto space C_{PP}. The matrix C is applied to each RGB value in $\bar{\mathbf{x}}_{\text{sRGB}}$.

$$C = C_{\text{PP}}\ C_{\text{CAT02}}\ C_{\text{sRGB}}^{-1} \tag{4.2}$$

This linear mapping converts all original in-gamut color values in a colorimetrically accurate manner. However, the colors of pixels initially clipped to the sRGB gamut, are constrained at the boundary of the sRGB gamut in the ProPhoto space. We referred to this transformed image as $\tilde{\mathbf{x}}_{\text{PP}}$, which is the output of the conventional approach for gamut expansion, known as *Clip*. Our goal is to process these out-of-gamut pixels to restore them to their original wide gamut values.

Out-of-gamut pixels Since the input sRGB image $\mathbf{x}_{\text{sRGB}}$ was generated through a conversion from its original ProPhoto encoding, we classify the pixels in $\mathbf{x}_{\text{sRGB}}$ into two classes: (1) the *in-gamut* pixels that were not clipped because they are inside the sRGB gamut; and (2) the possibly *out-of-gamut* pixels that have been potentially clipped down.

When the color of a pixel is on the sRGB gamut's boundary, the color may either be initially at the boundary or be clipped to the boundary. Therefore, these pixels are regarded as potentially out-of-gamut. We consider the pixels with one or more possibly saturated values in their three RGB channels as potentially out-of-gamut, with two exceptions of pure white and pure black colors, which are preserved during the color space conversion. On the other hand, all non-saturated pixels—pixels with color values in the exclusive range $(0, 255)$—are considered in-gamut.

Based on this simple definition, we compute a binary *out-of-gamut mask* for each input image to let our gamut mapping network focus on learning how to restore the out-of-gamut pixels.

The out-of-gamut mask $\mathbf{m}$ is computed based on the input sRGB image $\mathbf{x}_{\text{sRGB}}$ as

$$\mathbf{m} = \begin{cases} 0 & \text{where } \mathbf{x}_{\text{sRGB}} \text{ is black, white, or in-gamut} \\ 1 & \text{otherwise.} \end{cases} \tag{4.3}$$

The task of our network is to restore the out-of-gamut colors of pixels in the sRGB space to their original color in the ProPhoto space with the help of the in-gamut neighbors. Since we have the colorimetrically correct conversion for in-gamut pixels, our network focuses on estimating the amount of the saturation of each pixel instead of its original color, so we can add the estimated saturation (or residual) to the linearly mapped ProPhoto image to get the restored image—that is:

$$\hat{\tilde{\mathbf{y}}}_{\text{PP}} = \mathbf{m} \odot f_\theta(\tilde{\mathbf{x}}_{\text{PP}}, \text{inv}(\mathbf{m})) + \tilde{\mathbf{x}}_{\text{PP}}, \tag{4.4}$$

where f_θ is our network, called *GamutNet*, with parameters θ, $\tilde{\mathbf{x}}_{\text{PP}}$ is the input image linearly mapped to the ProPhoto space, $\odot$ indicates point-wise masking operation, and $\text{inv}(\cdot)$ indicates binary mask inversion.

Given a dataset of N image pairs

$$\mathcal{D} = \left\{ \left(\tilde{\mathbf{x}}_{\text{PP}}^{(i)}, \bar{\mathbf{y}}_{\text{PP}}^{(i)} \right) \right\}_{i=1}^{N} , \tag{4.5}$$

we train our neural network by minimizing the L_1 loss between the estimated linear ProPhoto image and its ground-truth counterpart, over the out-of-gamut pixels

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \left| \mathbf{m}^{(i)} \odot \left(\hat{\bar{\mathbf{y}}}_{\text{PP}}^{(i)} - \bar{\mathbf{y}}_{\text{PP}}^{(i)} \right) \right| , \tag{4.6}$$

where $|\cdot|$ indicates the L_1 norm. Fig. 4.2 shows our gamut mapping framework using a deep neural network, *GamutNet*.

4.3 Network Architecture and Training

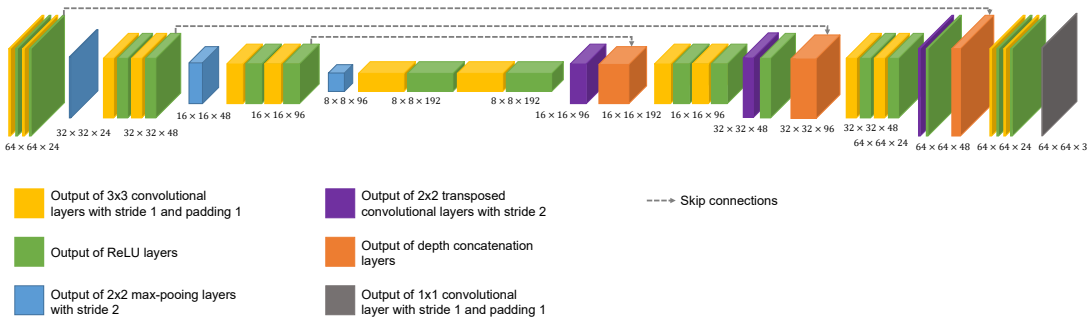


Figure 4.3: An illustraion of our deep neural network for gamut restoration.

Our deep network's architecture is based on U-Net [96] (see Figure 4.3), which has been shown more effective than a basic encoder-decoder as it makes low-level features available across the network [42]. We designed our DNN network to predict the residual between our input clipped-ProPhoto ($\tilde{\mathbf{x}}_{\text{PP}}$) image and the target image ($\bar{\mathbf{y}}_{\text{PP}}$). This means the residual represents the color difference between the linear-sRGB colors transformed into the ProPhoto color space with a clipped signal and its ProPhoto RGB counterpart (the

original signal). To help the network focus on restoring the out-of-gamut-pixels' values only with the in-gamut pixels' aid, we concatenate the inverted out-of-gamut mask to the input image as illustrated in Figure 4.2.

The out-of-gamut mask is also applied during the loss calculation to prevent the in-gamut pixels from contributing to the loss. This is because colorimetrically, the in-gamut pixels in the input image are almost identical to their counterpart in the ground-truth image and yield negligible loss. Using the out-of-gamut mask, we can filter out such trivial cases and focus our model on restoring out-of-gamut color values in the wider-gamut ProPhoto color space. We use L_1 loss for optimization, which is commonly used in many image restoration problems. We optimize our model using Adam [52] with an initial learning rate of 10^{-4}.

We train our model in a patch-based approach. We sample small patches from the full-size images. For each data pair, patches in the specified size are randomly chosen within the maximum number of patches per image. Considering that in-gamut pixels tend to out-number the opposite, we bias the patches' distribution toward having more out-of-gamut data. This is done by making patches centered at out-of-gamut pixels (i.e., saturated pixels) using the out-of-gamut mask.

A few more steps are required to generate a final output ProPhoto prediction. Given an output image computed feed-forwarding a prepared input through the train GamutNet model, we first take care of in-gamut pixels. Specifically, we replace the colors of in-gamut pixels with those of the input linear ProPhoto image. This simple process ensures that the in-gamut pixels, which are colorimetrically correct, stay intact. So far, the output colors are in linear color space; we apply the ProPhoto gamma function, followed by 8-bit quantization.

Finally, we save the final output image in the PNG format, embedding an ICC profile

denoting that the image has been saved in the ProPhoto color space. Embedding the ICC profile is required to allow the PNG image to be interpreted in the correct color space by software such as Photoshop.

4.4 Experiments and Results

4.4.1 Dataset

To produce our dataset, we first gather a large number of camera RAW images to be processed through a software camera emulator. For this, we use RAW images from the MIT-Adobe FiveK [16], NUS [21], Cube+ [11], and RAISE [23] datasets. These datasets collectively represented 16,599 images captured from different cameras and of a wide variety of scene content.

We use the Adobe Camera RAW SDK to serve as our software ISP as done in [48]. The Adobe Camera RAW SDK mimics the steps of a typical ISP. We can modify the SDK to output the 8-bit ProPhoto RGB after image enhancement and the final 8-bit sRGB image. Access to these internal image encodings in the SDK is detailed in [48]. The image enhancement steps shown in Fig. 2.10 control the photo-finishing applied to the image using a combination of a 3D lookup table and a 1D lookup table. These combined lookup tables form a particular "picture style." In this chapter, we use both the *Adobe Vivid* picture style and *Adobe Standard* picture style to render the images. We can think of this as a virtual camera that essentially integrates these two picture styles into a meta style. In the end, we generated 33,198 pairs of input (sRGB) and target (ProPhoto) images.

Before training, we need to exclude images that contain only a few out-of-gamut pixels. We also want to avoid images that result in overly saturated sRGB (i.e., most of the colors were out of gamut). Images with only a few out-of-gamut pixels do not provide sufficient

information to train our model. Similarly, overly saturated images force the network to learn without in-gamut neighbors' information. To cull such images, we compute the ratio p of the number of the out-of-gamut pixels to the total number of pixels of each image:

$$p = n_s \ / \ (n_g + n_s) \ , \tag{4.7}$$

where n_s and n_g are the number of the out-of-gamut pixels and that of the in-gamut pixels of the image, respectively. After excluding images with p greater than 50%, we select 5,000 images with the largest p to form our final dataset. We randomly split the 5,000 images into training, validation, and testing sets of 3,000, 1,000, and 1,000 images, respectively.

4.4.2 Results

We first evaluate our method's ability to perform gamut restoration from sRGB to ProPhoto. The evaluation uses the testing partition of our dataset described in the "Dataset" section. Note that all errors are reported only on the out-of-gamut colors.

ProPhoto Gamut Restoration

For our first evaluation, we compare our method with three representative approaches for color space conversion. The first is the recent gamut-extension method by Zamir et al. (2019) [115] that we present earlier in Chapter 3. We again note that the goal of Zamir et al. (2019)'s method *is not* accurate colorimetric reproduction. The second approach we compare with is Adobe Photoshop's color conversion utilities. In particular, we use Adobe Photoshop's *absolute* and *relative* colorimetric color conversion utilities. These represent the most common strategy found in other software or display devices to minimize colorimetric errors. The difference between the two approaches is that for the relative colorimetric approach, the white-point is shifted slightly from D65 to D50 to

Table 4.1: This table shows the RMSE and ΔE_{00} results of color space conversion between sRGB and ProPhoto on 1,000 images with known ProPhoto color values. Methods used are Zamir et al. gamut-expansion, Photoshop's color conversion feature (absolute and relative colorimetric), Clip, and our GamutNet result.

	RMSE↓				ΔE_{00}↓			
Methods	**Mean**	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**
Zamir et al. [115]	9.533	4.405	6.730	10.127	6.101	3.836	5.720	8.315
Photoshop, Absolute	5.892	2.475	3.958	5.927	3.509	2.293	3.111	4.106
Photoshop, Relative	5.891	2.475	3.958	5.927	3.509	2.293	3.111	4.106
Clip	5.289	2.083	3.383	5.243	3.244	2.061	2.861	3.867
GamutNet	**2.812**	**1.319**	**1.719**	**2.500**	**2.209**	**1.141**	**1.611**	**2.514**

Table 4.2: Results targeting Display P3 after conversion using Photoshop (Relative colorimetric). Our method yields the best results in terms of RMSE and ΔE_{00}.

	RMSE↓				ΔE_{00}↓			
Methods	**Mean**	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**
Zamir et al. [115] (Direct)	13.428	6.875	9.985	13.573	7.133	5.090	6.870	8.945
Zamir et al. [115]	11.856	6.489	9.367	12.664	6.866	5.254	6.764	8.570
Photoshop, Absolute	10.904	4.819	8.198	12.065	5.960	3.602	5.030	7.203
Photoshop, Relative	10.903	4.819	8.197	12.064	5.959	3.601	5.030	7.202
Clip	9.769	4.006	7.085	10.827	5.665	3.695	5.124	6.814
GamutNet	**4.282**	**2.189**	**3.023**	**4.385**	**2.889**	**1.964**	**2.450**	**3.313**

Table 4.3: Results targeting Adobe RGB after conversion using Photoshop (Relative colorimetric). Our method yields the best results in terms of RMSE and ΔE_{00}.

	RMSE↓				ΔE_{00}↓			
Methods	**Mean**	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**
Zamir et al. [115] (Direct)	11.363	5.159	6.980	9.943	5.982	4.158	5.385	7.103
Zamir et al. [115]	10.034	4.969	6.677	9.237	5.840	4.235	5.535	7.101
Photoshop, Absolute	10.343	5.474	8.633	11.749	8.018	6.170	7.873	9.411
Photoshop, Relative	10.342	5.473	8.633	11.748	8.017	6.170	7.872	9.410
Clip	8.372	4.281	6.732	9.387	6.731	5.152	6.774	8.084
GamutNet	**3.935**	**2.365**	**2.824**	**3.466**	**3.014**	**1.713**	**2.287**	**3.446**

match the white-point definition in ProPhoto. The white-point shift has little effect on color values in the final mapped image.

Finally, for completeness, we also show the results of the conventional 3×3 color space transform from linear-sRGB to linear-ProPhoto, which is called *Clip* (see Section 4.2).

Table 4.1 shows the comparison of all methods in terms of root mean squared error (RMSE) and ΔE_{00} computed over the out-of-gamut pixels. ΔE_{00} is a color-based metric that accounts for non-uniformity of perceptual color appearance. The lower the ΔE_{00} score, the more similar colors are considered perceptually. The table shows the mean score over the 1,000 test images, the top 25% quantile, the 50% quantile (median), and the bottom 25% quantile for both RMSE and ΔE_{00}. Our GamutNet approach provides much better results for all metrics.

Subjective results are shown in Figure 4.4, Figure 4.5, Figure 4.6, Figure 4.7, Figure 4.8 and Figure 4.9. Note that for each chapter, the figures with qualitative results are shown at the end of the chapter. We also include results from Apple's *ColorSync* software [41]. Unlike Photoshop, we were not able to batch process the 1,000 test images with ColorSync. As a result, we included examples from ColorSync for visual comparisons only in our figures and not in Table 4.1. In our qualitative results, the figures show several images each with the following: input sRGB; Photoshop (relative colorimetric) conversion; ColorSync's conversion; our GamutNet conversion; and (5) the ground-truth ProPhoto target. Below each image, we show a heat map of the RMSE error ranging from [0-20] and a CIE-xy chromaticity diagram showing the out-of-gamut sRGB colors after conversion. We can see that GamutNet gives the best quantitative results on both metrics. The chromaticity diagram also reveals that GamutNet shifts the out-of-gamut colors to be much more similar to their ground-truth positions.

Conversion to Adobe RGB and Display P3

GamutNet can be used to map to other color spaces. In color management systems, an ICC color profile describes how a color space can be first mapped to a wide-gamut profile connection space (PCS) and then from this space to other spaces. The PCS is normally CIE XYZ; however, we can use our ProPhoto space as a proxy for CIE XYZ.

To evaluate this task, we convert the 1,000 ground-truth ProPhoto test images to Adobe RGB and Display P3 using Adobe Photoshop. We then compare our ability to map to Adobe RGB and Display P3 against Photoshop as done in the prior section. We also compare our results with Zamir et al.'s method [115]. For Zamir et al.'s method, we perform two approaches. The first is similar to ours, where the sRGB image is converted to ProPhoto, and then gamut mapped to Adobe RGB and Display P3 using Photoshop. The results are shown in Figure 4.10, Figure 4.11, Figure 4.12 and Figure 4.13. We also show Zamir et al.'s method applied to expand the sRGB input to Adobe RGB and Display P3. These results are denoted as Zamir (Direct). Results are again reported in terms of RMSE and ΔE_{00} for the out-of-gamut pixels. GamutNet provides close to 50% reduction in errors as shown in Table 4.2 and Table 4.3.

4.5 Summary

We have presented our deep-learning-based method for wide-gamut restoration. Our approach takes advantage of the fact that camera pipelines process images internally in the wide-gamut ProPhoto color space. This allowed us to generate a dataset with more than 5,000 pairs of images in both ProPhoto RGB and sRGB color spaces for training and testing our deep model. Our results show the deep network is able to successfully restore many of the original ProPhoto RGB values, providing several dB improvements in terms

of PSNR. We also showed how this approach is useful for conversion to other color spaces, such as Adobe RGB and Display P3.

Like any other deep learning approaches that employs complex architectures such as U-net, our method, GamutNet, faces certain limitations. One significant challenge is it requires large amounts of labeled training data to achieve high accuracy. Additionally, our GamutNet can struggle with generalization to images not represented in the training set, such as those pre-processed with different photo-finishing styles, leading to reduced performance on unseen or novel data. Especially in our case of gamut restoration, GamutNet is designed to predict the residuals of OOG values based on the context of in-gamut pixels. It falls short when processing images with a high proportion of OOG pixels, such as those that are overly saturated, limiting its applicability in these scenarios.

In the forthcoming chapters, we will introduce methods for gamut restoration designed to overcome these limitations.

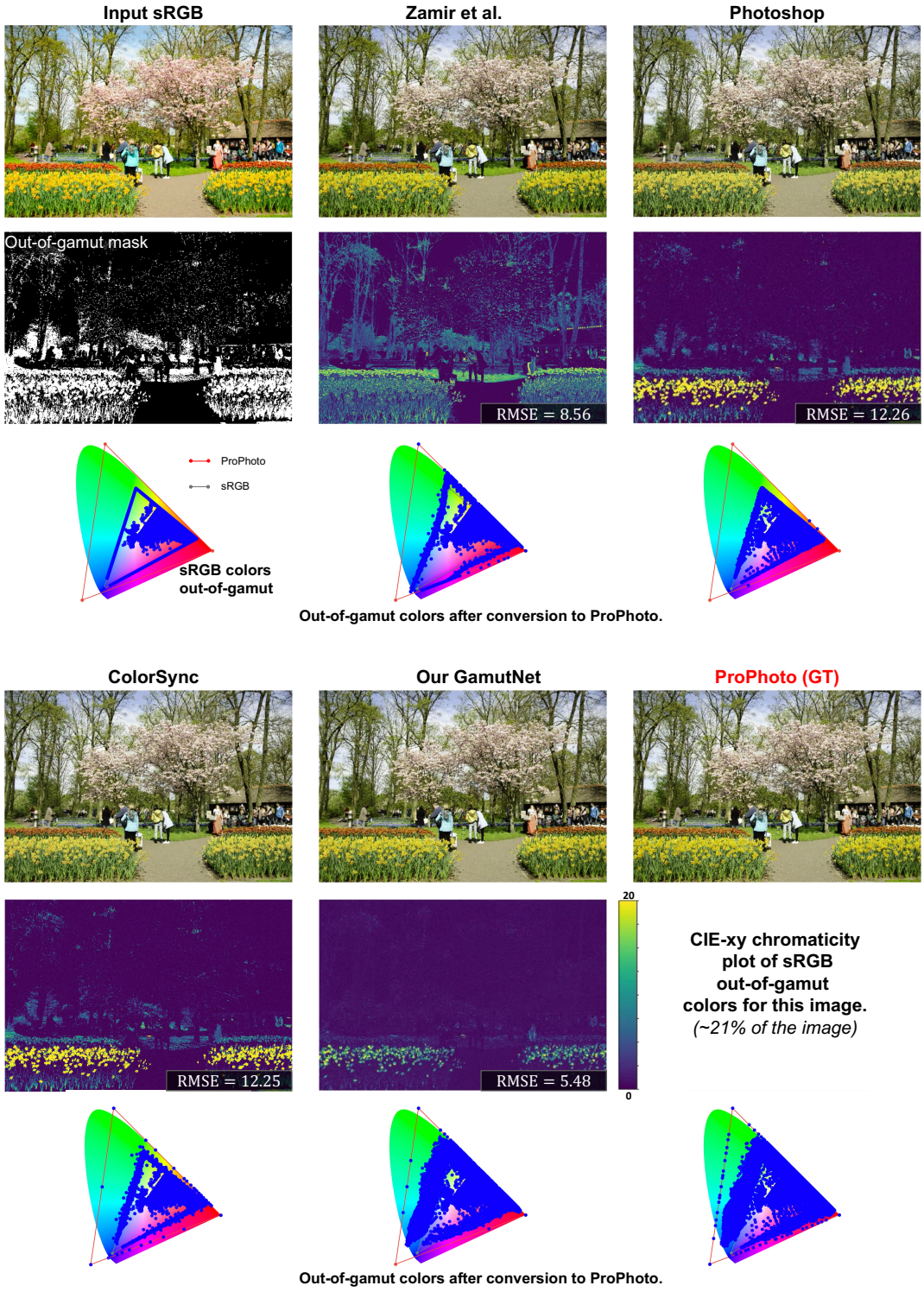


Figure 4.4: Comparisons among the ProPhoto results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto.Photoshop and ColorSync are applying similar colorimetric conversion. The GamutNet provides the best color space conversion.

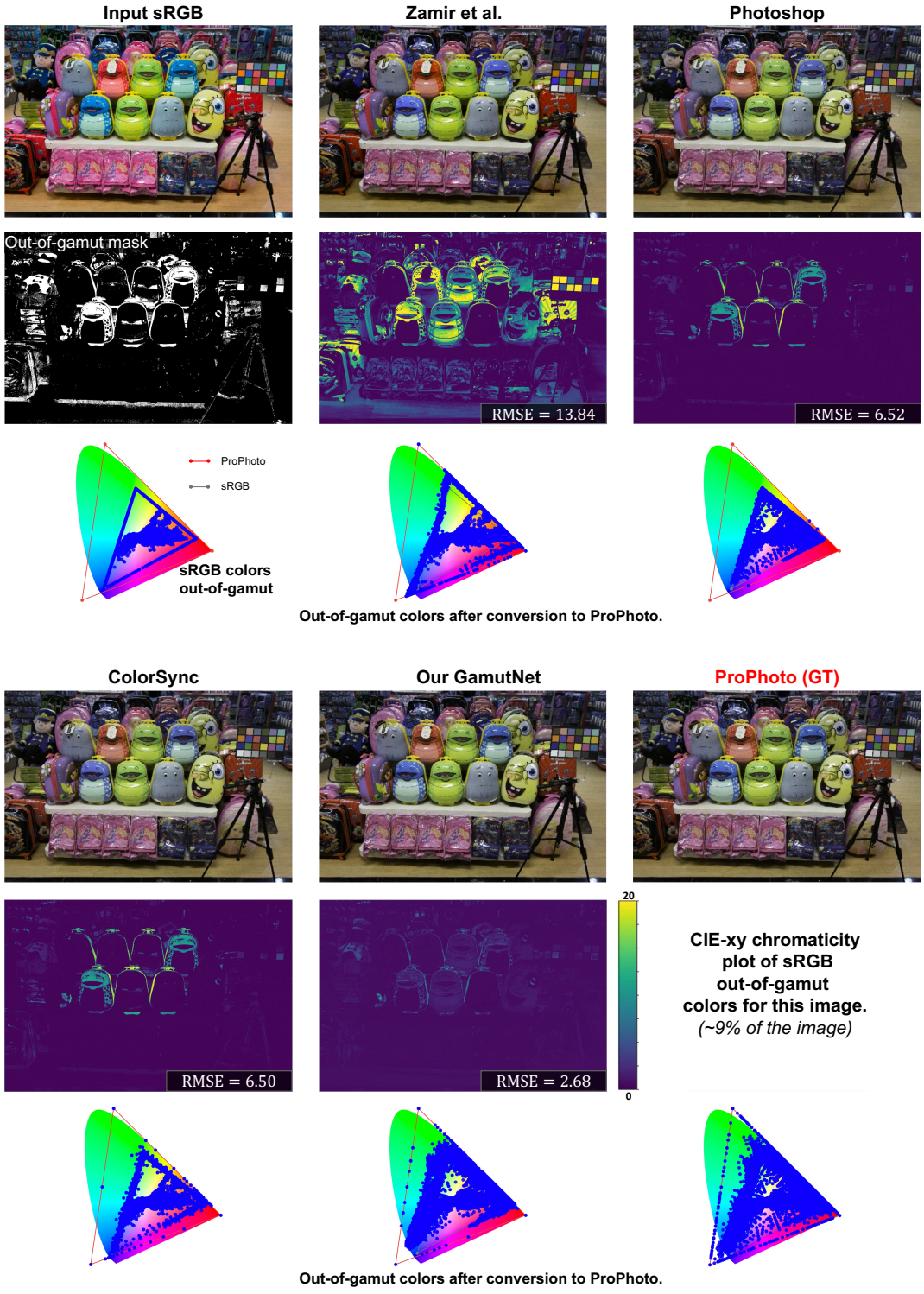


Figure 4.5: Comparisons among the ProPhoto results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto. Photoshop and ColorSync are applying similar colorimetric conversion. The GamutNet provides the best color space conversion.

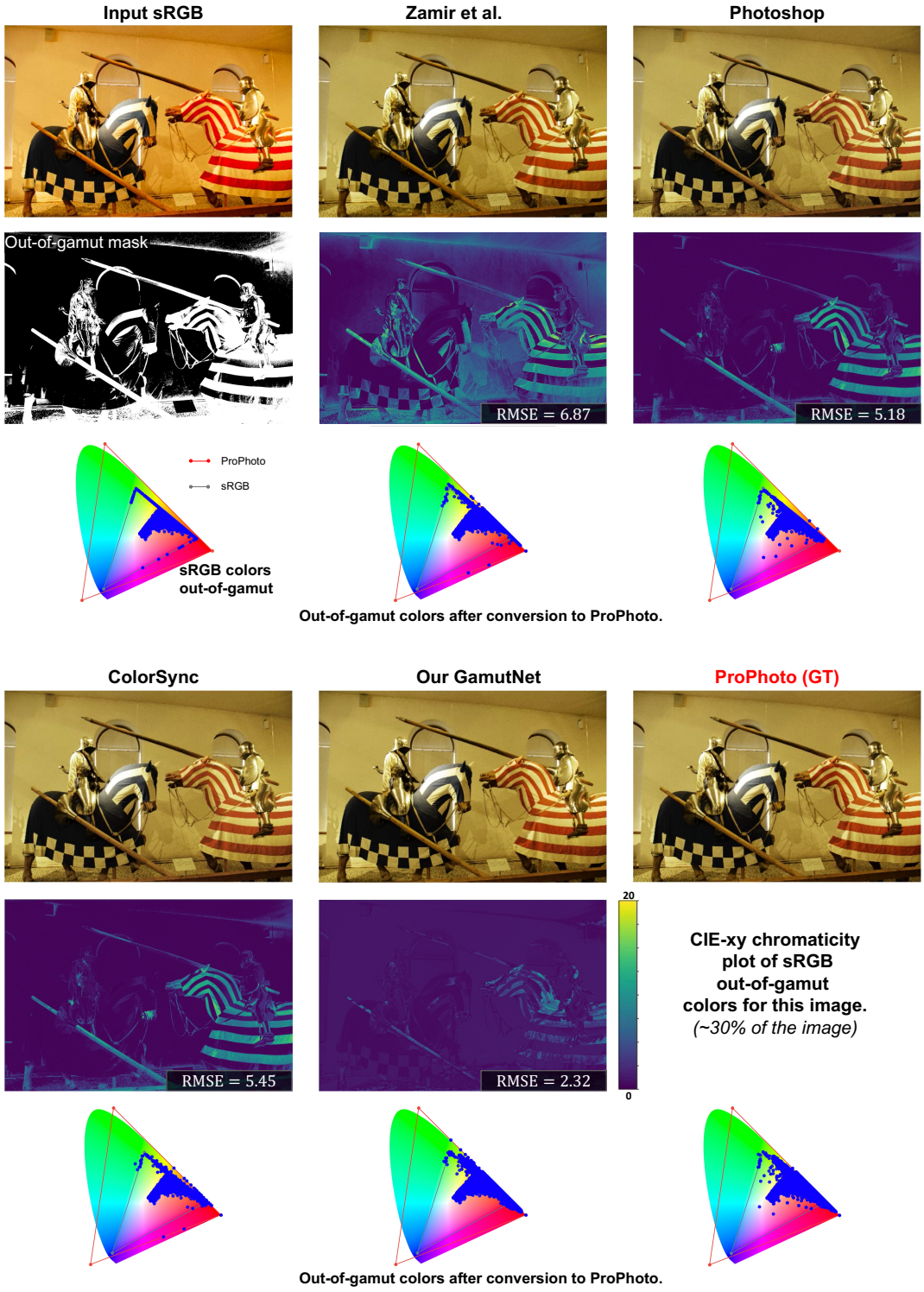


Figure 4.6: Comparisons among the ProPhoto results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto. Photoshop and ColorSync are applying similar colorimetric conversion. The GamutNet provides the best color space conversion.

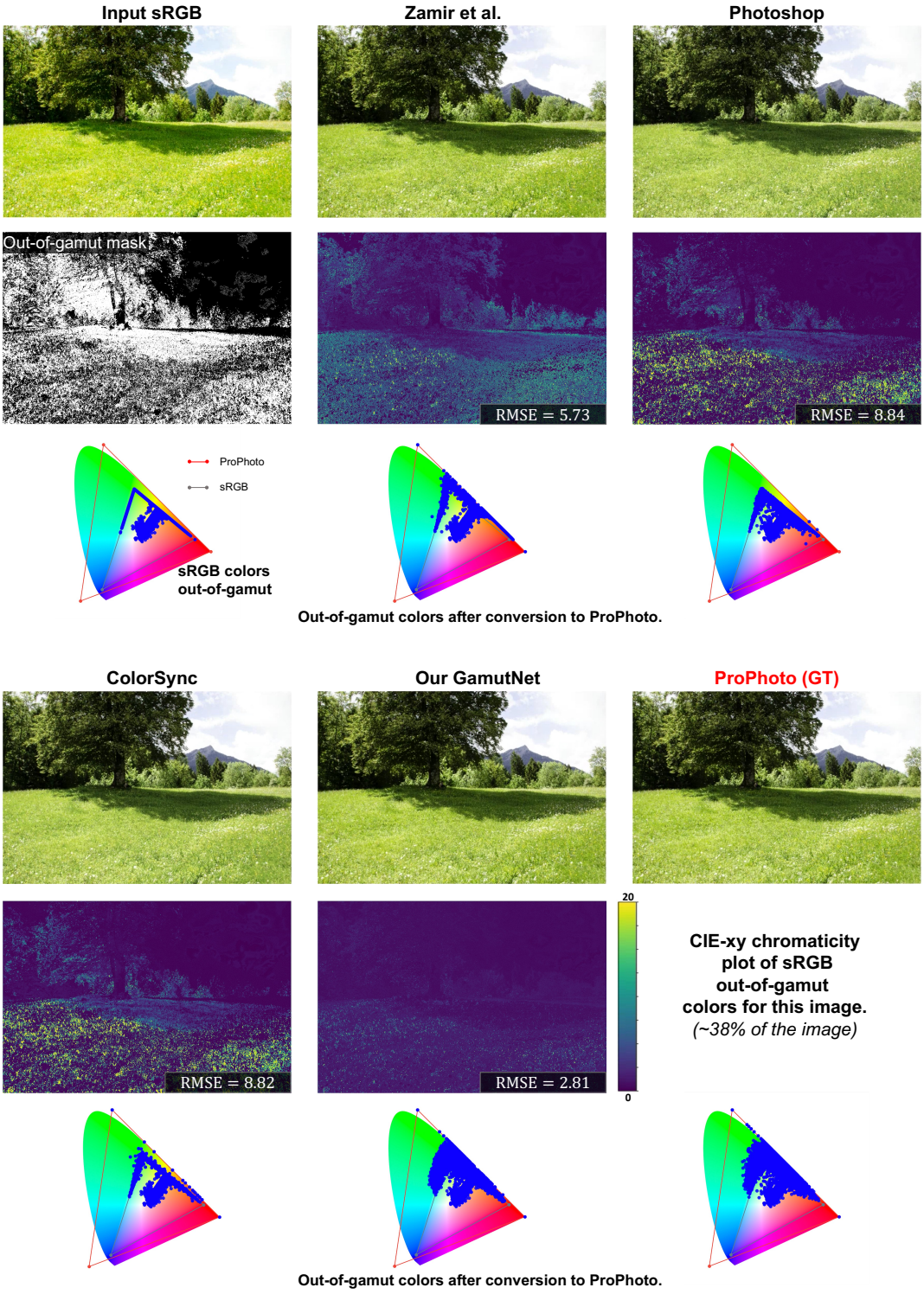


Figure 4.7: Comparisons among the ProPhoto results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto. Photoshop and ColorSync are applying similar colorimetric conversion. The GamutNet provides the best color space conversion.

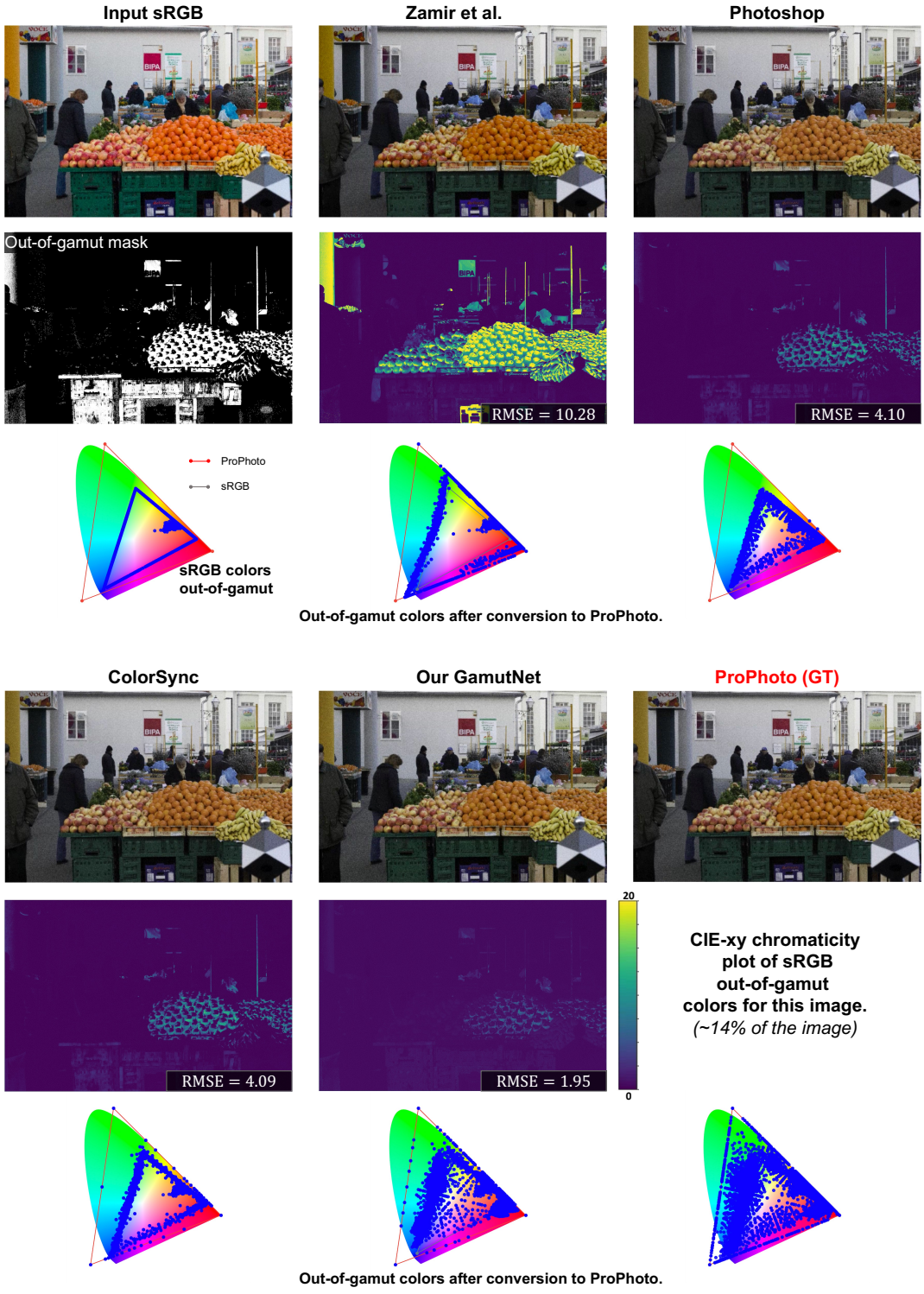


Figure 4.8: Comparisons among the ProPhoto results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto. Photoshop and ColorSync are applying similar colorimetric conversion. The GamutNet provides the best color space conversion.

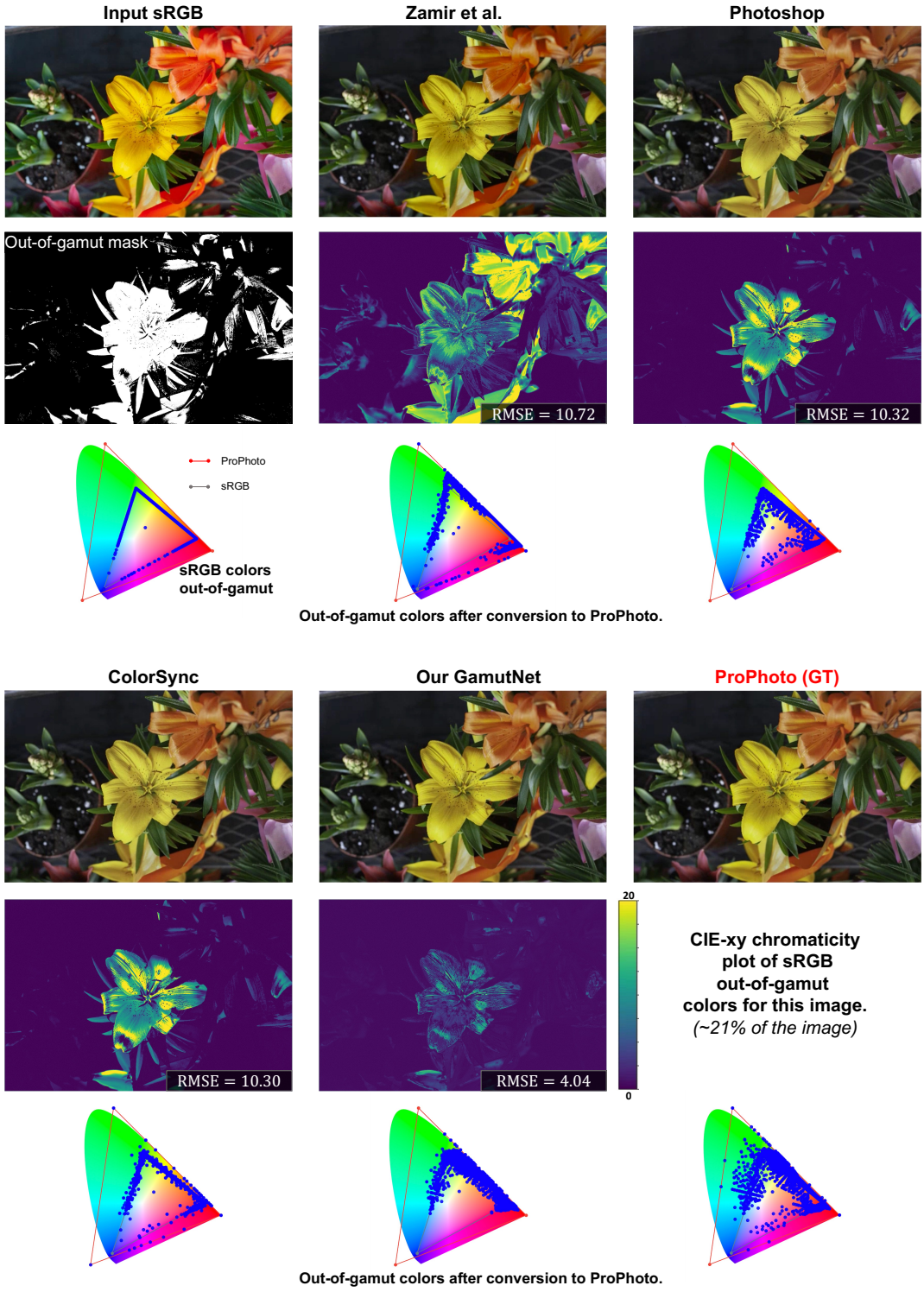


Figure 4.9: Comparisons among the ProPhoto results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto. Photoshop and ColorSync are applying similar colorimetric conversion. The GamutNet provides the best color space conversion.

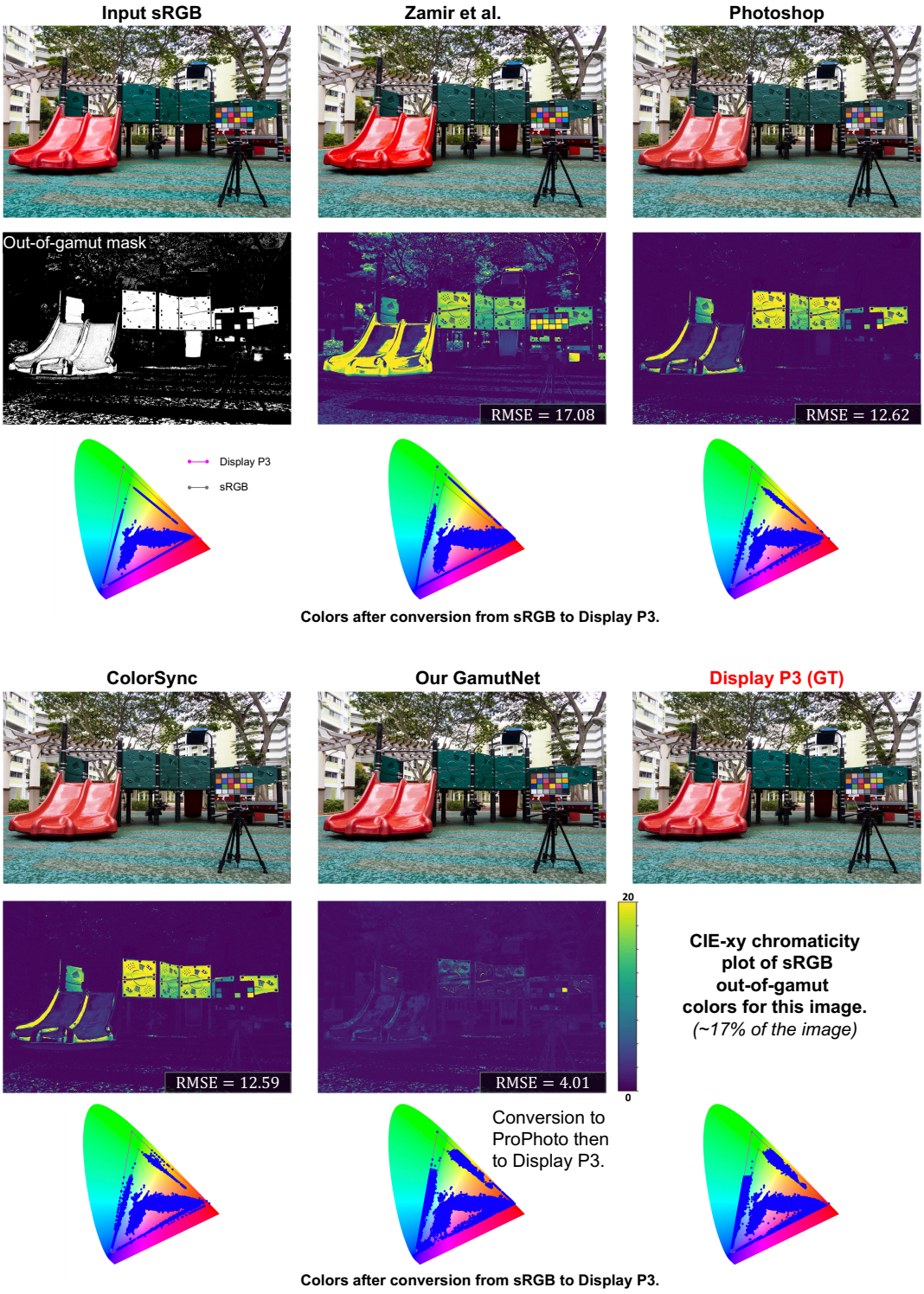


Figure 4.10: Comparisons among the Display P3 results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB, and Display P3. GamutNet provides the best color space conversion.

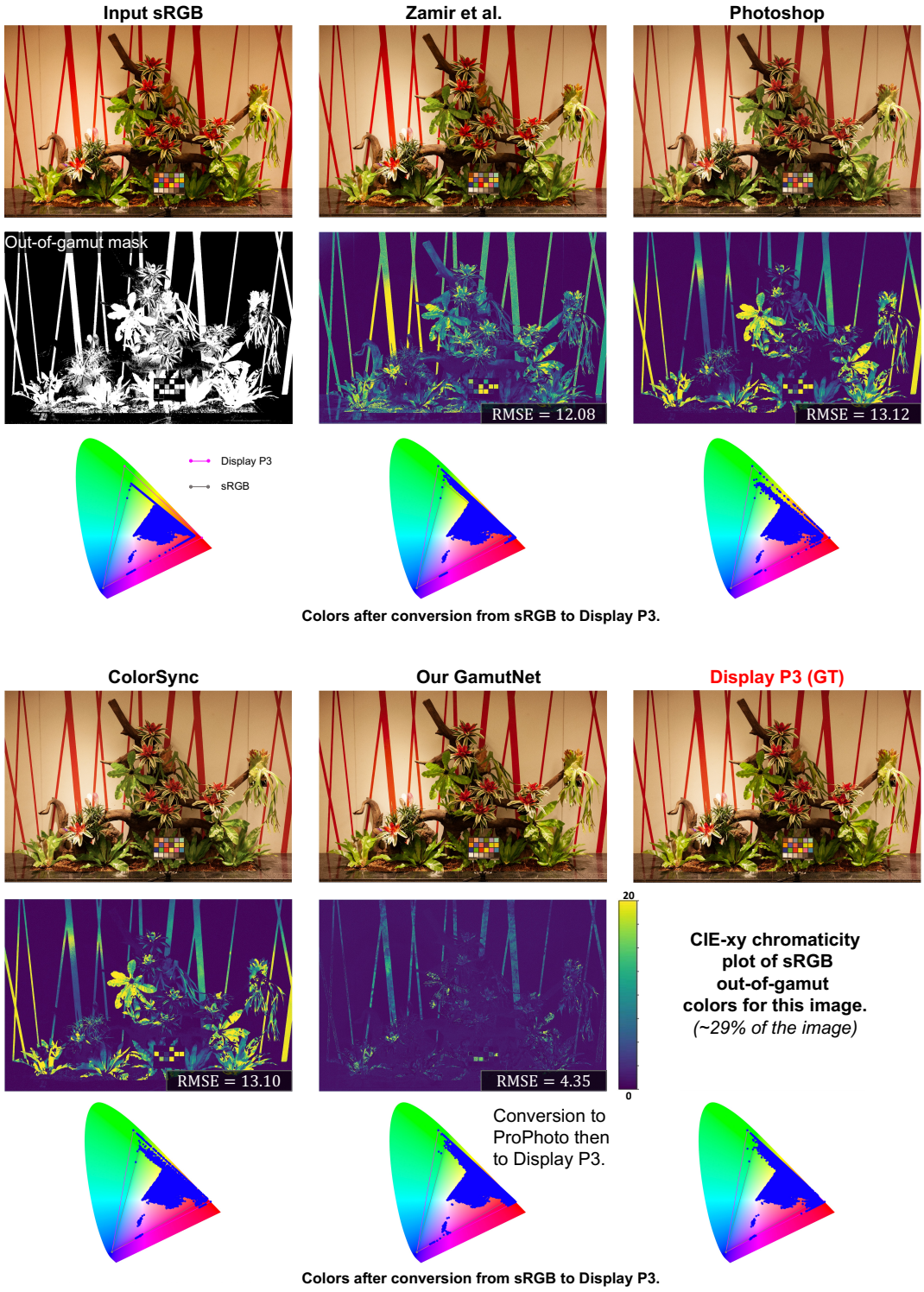


Figure 4.11: Comparisons among the Display P3 results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB, and Display P3. GamutNet provides the best color space conversion.

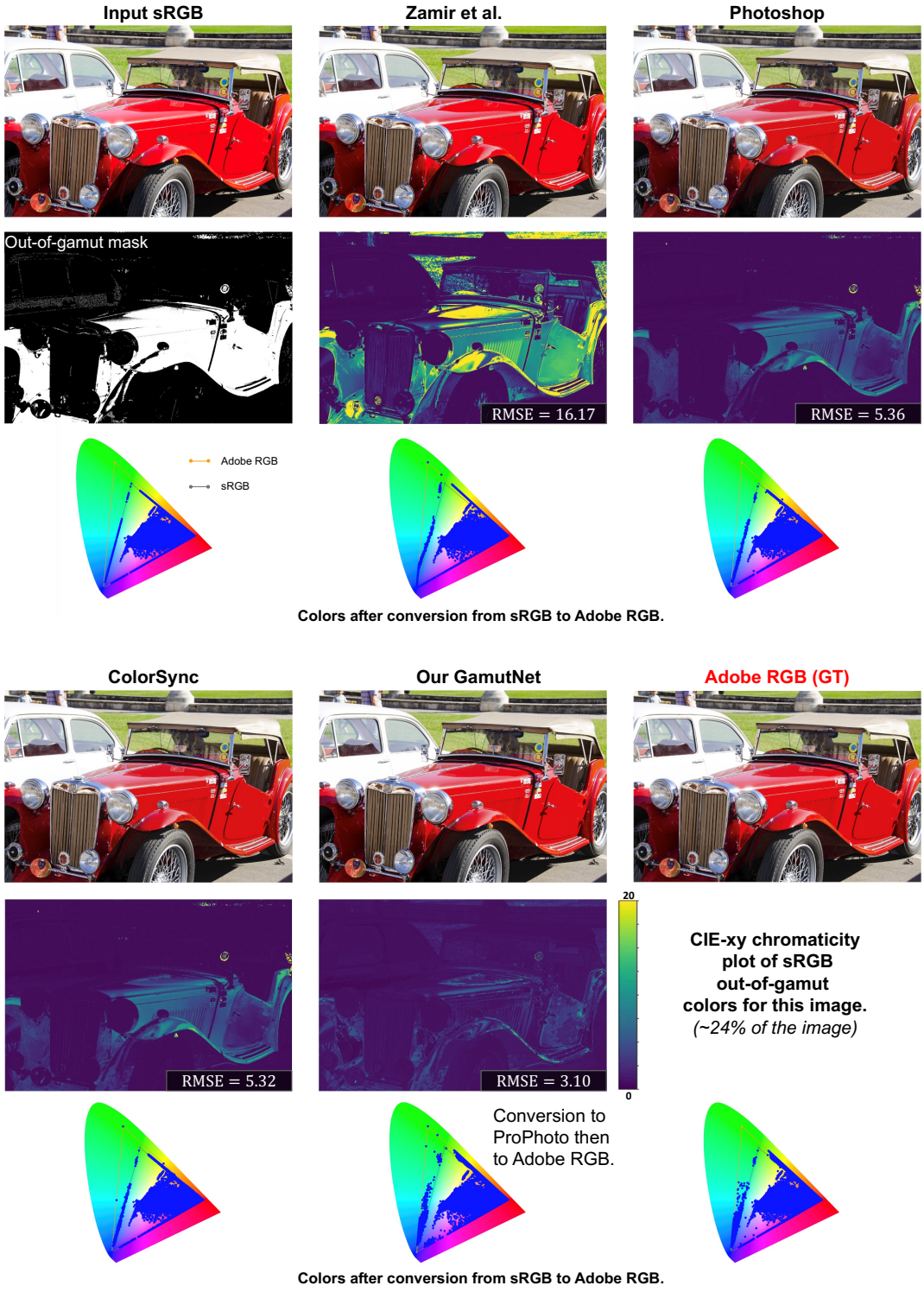


Figure 4.12: Comparisons among the Adobe RGB results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB, and Adobe RGB. GamutNet provides the best color space conversion.

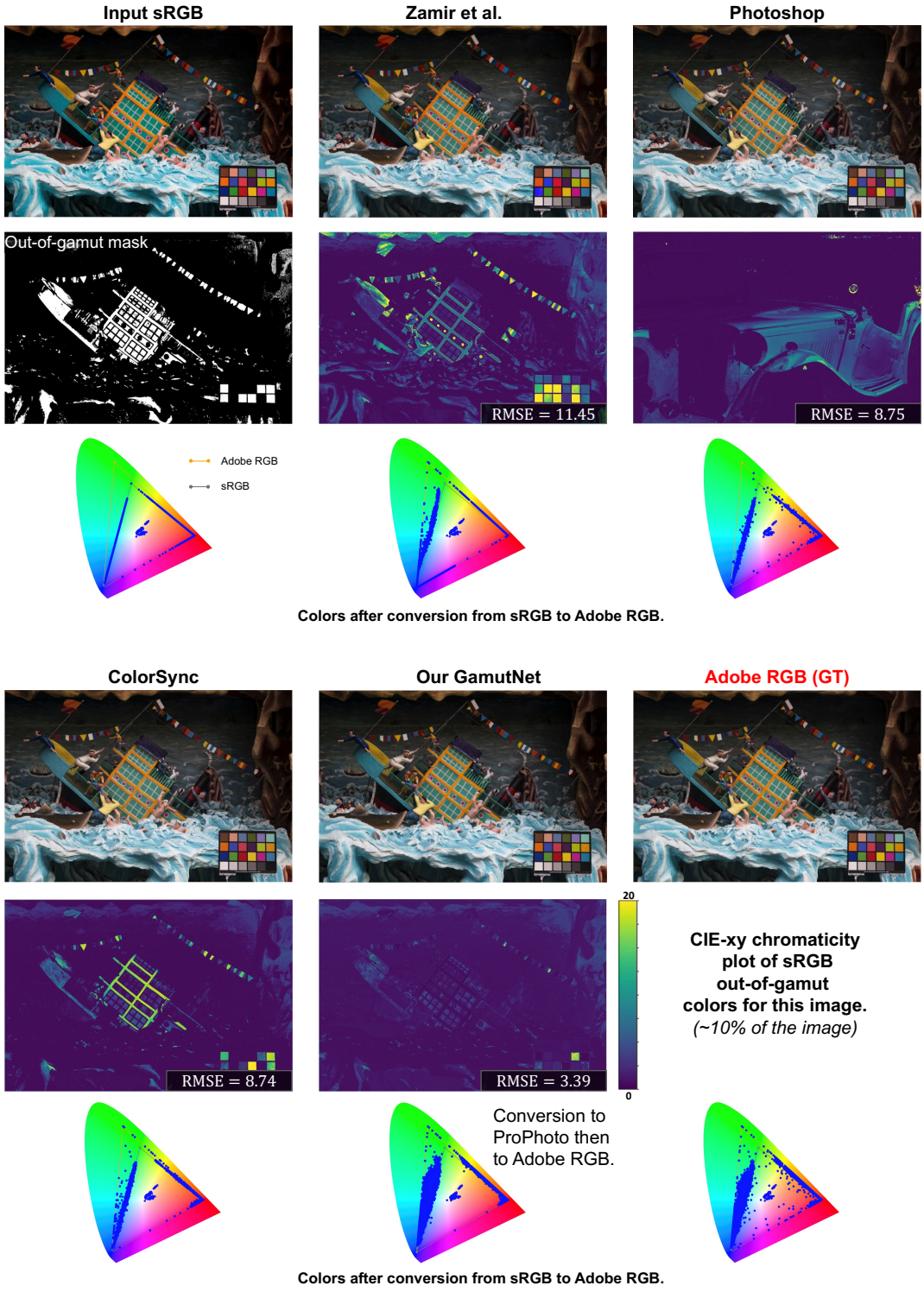


Figure 4.13: Comparisons among the Adobe RGB results of Zamir et al. [115] (gamut expansion), Photoshop [40], ColorSync [41], and our GamutNet. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB, and Adobe RGB. GamutNet provides the best color space conversion.

5 Gamut Restoration via Image Samples

In this chapter, we describe our second method for recovering wide-gamut colors that use specialized metadata sampled at capture time. We first give an overview of our metadata-assisted method, followed by a description of two different mapping functions (global and local) that can be used to recover the wide-gamut color values. Finally, we explain our experiments and results. The source code and dataset for this work can be found here: `https://github.com/hminle/improving-color-space-conversion-via-metadata`

5.1 Introduction

As we present in Chapter 3, the onboard camera rendering hardware includes several steps that convert the RAW sensor image to its final display-referred output. One of these steps involves the conversion of the scene-referred RAW image to a wide-gamut reference output medium metric (ROMM) color space [19,20] commonly referred to as ProPhoto. ProPhoto contains around 90% of the visible colors, which is used as a profile connection space. Inside the camera pipeline, photo- finishing routines are applied to the ProPhoto image to improve its photographic qualities. The final stage in the camera pipeline is to convert the ProPhoto/ROMM image to a display-referred color space (e.g., sRGB, AdobeRGB, Display-P3).

Several works proposed to embed metadata to allow the image to be reverted from a

display-referred color space (i.e., sRGB) back to its scene-referred RAW image state [80,82, 83,92]. However, reverting back to RAW would require a forward rendering to convert the RAW into a output-referred color space. The necessary data to re-render the RAW image correctly is rarely available. Our ProPhoto encoding does not suffer from this problem as it directly represents the rendered image's colors. We also note recent work in [5] that embedded metadata regarding the image's color rendering options (i.e., white-balance) to allow post-capture white-balance manipulation. Similar to [5], we found that only a small amount of color samples is required to assist in our gamut mapping.

We introduce a method designed to enhance gamut restoration by embedding metadata of original wide-gamut values directly into the captured image. Our observation comes from the fact that most camera hardware applies a conversion of the original sensor-RGB image to an intermediate image state in the wide-gamut ProPhoto color space when rendering the image to the final display-referred color space. We demonstrate that by embedding just a very small number of these wide-gamut ProPhoto samples (for instance, 150×150 samples in a 20MB image), it is possible to develop global color mapping functions that significantly improve the accuracy of wide-gamut restoration. Our technique addresses and resolves the generalization challenge encountered by GamutNet, as discussed in Chapter 4, through the use of original wide-gamut value metadata for enhanced gamut restoration. Additionally, we explore how this strategy can be expanded to incorporate local color mapping, further refining the restoration process. This method not only capitalizes on the intrinsic data available from the camera's capture process but also provides a scalable solution to overcoming common limitations in gamut restoration practices.

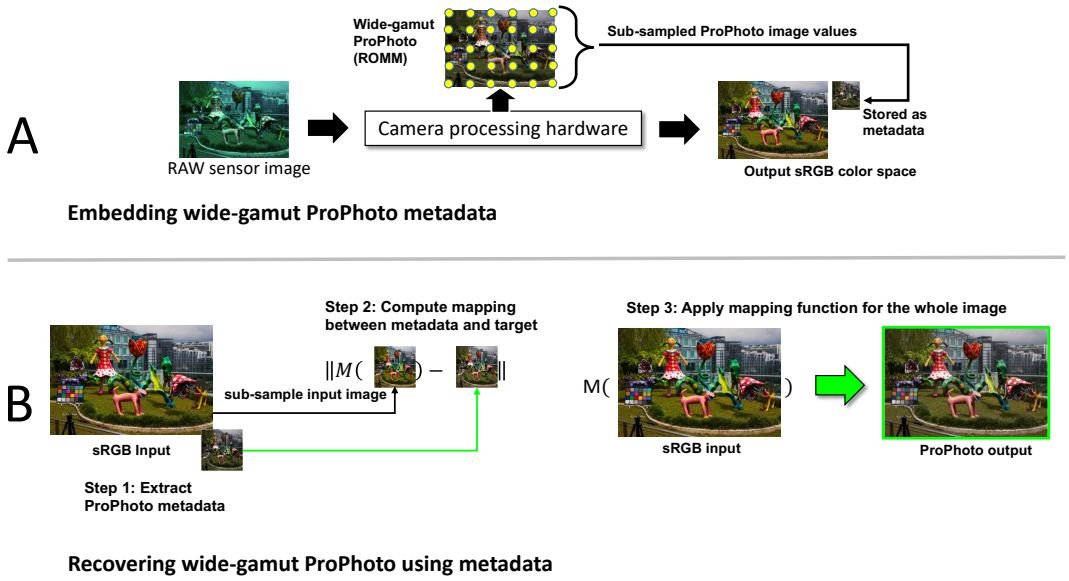


Figure 5.1: Overview of the second proposed method. (A) The camera hardware converts the initial RAW sensor image into an intermediate wide-gamut color space (e.g., ProPhoto color space) format for photographic rendering purposes. Our method extracts a small sub-sampling of the ProPhoto image (150$times$150 samples) and stores this informatoin as a metadata in the final saved standard color space (e.g., Adobe RGB (1998), Display P3, or sRGB). (B) To recovering wide-gamut ProPhoto values, we first retrieve our sampled wide-gamut image from the metadata stored in the final camera-rendered image. Then, we compute the mapping between the sub-sample of input image with the sampled wide-gamut image. Finally, we apply a locally adaptive nonlinear mapping to map from the sRGB input to ProPhoto output.

5.2 ProPhoto Sub-Sampling

Our second method involves embedding a small amount of wide-gamut image data in the final outputted image saved in a medium- or small-gamut color space. These wide-gamut

samples are gleaned from the camera ISP at capture time. This wide-gamut metadata is then used to compute color space conversion mappings for converting between different color spaces. An overview of our metadata-embedded gamut restoration method is shown in Figure 5.1.

As discussed in Chapter 2, the camera image processing hardware applies a series of processing steps to convert a captured RAW sensor image to its final rendered output. One of the steps performed is a mapping the image color values to the ProPhoto/ROMM color space (see [48] for an overview of in-camera processing). This ProPhoto color space provides a wide-gamut color encoding suitable for photo-finishing routines, such as general color enhancement, selective color rendering for skin, and local and global tone manipulation. When the photo-finishing is complete, the last step applied by the hardware is to convert the image to its final display-referred color space and then saved the file in a compressed format (i.e., JPEG or HEIC). DSLR cameras often allow the user to change the display-referred output color space (e.g., sRGB or Abobe RGB), while smartphone cameras typically use a fixed output color space (e.g., Display-P3 or sRGB). Our key insight is that before the image is saved, the image has pixel values encoded in wide-gamut color space. It is this wide-gamut ProPhoto data—available on the hardware for every captured image but later discarded–that we exploit. The idea of gleaning information from the camera hardware at rendering time and embedding it in the saved image is inspired by prior work [5, 83] that targeted different applications.

To this end, we propose a minimal modification to existing camera hardware to be able to extract the ProPhoto data as shown in Figure 5.1-(A). Specifically, we uniformly sub-sample a *tiny image* termed, $\mathbf{S}_{\text{ProP}}$, from the intermediate ProPhoto image during the photo-finishing process. This sampling is applied to the hardware's internal ProPhoto photo-finished image just before it is converted to its output color space encoding. There

are alternative ways to sub-sample the given image such as using histogram, but these methods are not hardware-friendly and require a considerable amount of processing time. In our experiments, we sample 150×150 pixels from the ProPhoto image, $\mathbf{I}_{\text{ProP}}$. We then store the ProPhoto samples, $\mathbf{S}_{\text{ProP}}$, as auxiliary data in the output image, $\mathbf{I}_{\text{sRGB}}$, where $\mathbf{I}_{\text{sRGB}}$ is encoded in sRGB color space. We choose sRGB as it is the most common standard color space, but our method can work with any other RGB color space that has gamut less than ProPhoto, namely Adobe RGB, or Display P3. The auxiliary ProPhoto data of 150×150 can be stored as an uncompressed comment field in an JPEG or HEIC file using just around 130KB for a full-size image (20MB).

5.3 Color Space Conversion with Wide-Gamut Metadata

Given a camera image $\mathbf{I}_{\text{sRGB}}$ with the stored metadata $\mathbf{S}_{\text{ProP}}$, the goal is to restore our sRGB image's colors to its wide-gamut representations. We describe two strategies—global and local—for this color space conversion in the following. Note that we assume images are represented in an $3 \times n$ matrix for format, where n refers to the total number of pixels in the image.

Global Mapping Our approach follows the strategy proposed in [5] of applying a global fitting to map our camera-rendered image $\mathbf{I}_{\text{sRGB}}$, in its full-size, to the target color space ProPhoto. For this procedure, we first extract the ProPhoto samples, $\mathbf{S}_{\text{ProP}}$, from our gamut reduced image $\mathbf{I}_{\text{sRGB}}$.

Next we followed the same sampling mechanism used in Sec. 3.1 to sample a tiny version of the $\mathbf{I}_{\text{sRGB}}$ source image. Here, we refer to this sampled image as $\mathbf{S}_{\text{sRGB}}$. Our global color conversion function can then be computed in a closed form as follows:

$$\arg \min_{\mathbf{M}_{\text{global}}} \left\| \mathbf{M}_{\text{global}} \, \Phi \left(\mathbf{S}_{\text{sRGB}} \right) - \mathbf{S}_{\text{ProP}} \right\|_{\text{F}}, \tag{5.1}$$

where $\|.\|_{\mathrm{F}}$ is the Frobenius norm and Φ is a polynomial function that maps the RGB triplets to a higher-dimensional space. In our implementation we used the polynomial mapping function used in [36]. Specifically, this polynomial function is represented as $\Phi : \Phi\left([\mathrm{R},\ \mathrm{G},\ \mathrm{B}]^{\top}\right) \rightarrow [\mathrm{R},\ \mathrm{G},\ \mathrm{B},\ \mathrm{RG},\ \mathrm{RB},\ \mathrm{GB},\ \mathrm{R}^2,\ \mathrm{G}^2,\ \mathrm{B}^2,\ \mathrm{RGB},\ 1]^{\top}$. Once the global fitting matrix $\mathbf{M}_{\text{global}} \in \mathbb{R}^{3\times 11}$ is computed, we can generate the reconstructed ProPhoto image $\hat{\mathbf{I}}_{ProP}$ as follows:

$$\hat{\mathbf{I}}_{ProP} = \mathbf{M}_{\text{global}}\ \Phi\left(\mathbf{I}_{\text{sRGB}}\right). \tag{5.2}$$

This procedure is shown in Figure 5.1-(B). In that example, the source image is saved in sRGB color space that has the smallest gamut. We then extract the ProPhoto metadata samples and compute the mapping $\mathbf{M}$, then apply the mapping to the whole source input image.

Local Mapping The global mapping described above achieves a noticeable improvement compared to conventional color space conversion methods; however, there is still an ambiguity in the one-to-many mapping when the target gamut has a wider color gamut (e.g., from the sRGB color gamut to the Display-P3 or ProPhoto color gamut). In such cases, a single color in the source gamut may have multiple corresponding colors in the target gamut. To minimize errors, the global mapping will map these source colors towards the mean of the corresponding target colors.

To overcome this, we can use a local mapping function that incorporates the spatial location (x, y) of a pixel in the mapping function. We perform a simple procedure to break the image into local regions and compute local mappings for each regions in a weighted least squares manner. First, the corresponding tiny source image, $\mathbf{S}_{\text{sRGB}}$, is segmented into superpixels using the simple iterative clustering (SLIC) algorithm [3] with the number of superpixels, k, set to 100. For each of these 100 superpixels, p, we build a binary mask

$\mathbf{L}_{(p)}$ where all pixels in $\mathbf{S}_{\text{sRGB}}$ that belong to p are set to 1, and 0 otherwise. This masks is dilated with a morphological operator and blurred to produce a weight map denoted as $\mathbf{L}_p$. This weight map is normalized so all of its weights sum to 1.

We then compute a mapping function per superpixel, p, as follows:

$$\arg\min_{\mathbf{M}_{\text{p}}} \left\| \left(\mathbf{S}_{ProP} - \mathbf{M}_p \, \Phi\left(\mathbf{S}_{\text{sRGB}}\right)\right)^{\top} \mathbf{W} \left(\mathbf{S}_{ProP} - \mathbf{M}_p \, \Phi\left(\mathbf{S}_{\text{sRGB}}\right)\right) \right\|_{\text{F}}, \tag{5.3}$$

where $\mathbf{W}$ is a diagonal weight matrix where each entry is the weight from the weight map. Eq. 5.3 represents a weighted least squares fit between the source $\mathbf{S}_{\text{sRGB}}$ and $\mathbf{S}_{\text{ProP}}$ for those pixels belonging to superpixel p.

The superpixel segmentation mask is upsampled to the size of the input image $\mathbf{I}_{\text{out}}$. Each pixel in $\mathbf{I}_{\text{sRGB}}$ belonging to the upsampled superpixel p is converted to $\hat{\mathbf{I}}_{\text{ProP}}$ using the mapping function $\mathbf{M}_p$ as described in Eq. 5.2. We find this simple local mapping procedure gives improved results. We can operate the entire process in just $\sim$1.5 minutes using unoptimized Matlab code for a full-frame 20MB image with a normal CPU.

5.4 Experiments and Results

5.4.1 Dataset

We use the same dataset that we already describe in the previous Chapter 4. In this experiment, we test our global and local mappings over 1,000 test images and compare them against Zamir et al. [115], Photoshop [40], the conventional clipping method, and our first approach, GamutNet.

5.4.2 Quantitative results

We report the average root mean square error (RMSE) and peak signal-to-noise ratio (PSNR) between predicted and ground truth test images. Metrics are provided for the

Table 5.1: This table shows results on various methods used for wide-gamut color recovery for 1,000 test images in full resolution. Shown are RMSE and ΔE_{00} for the out-of-gamut pixels in images. We compare our approaches with Zamir et al. [115], Photoshop (absolute and relative rendering intents) [40], and conventional clipping method 4.2.

Methods	**RMSE**↓	ΔE_{00}↓
Zamir et al. [115]	9.533	6.101
Photoshop, Absolute	5.892	3.509
Photoshop, Relative	5.891	3.509
Clip	5.289	3.244
GamutNet	2.812	2.209
ProPhoto-Sampled Global Mapping	**2.795**	**2.398**
ProPhoto-Sampled Local Mapping	**2.379**	**1.962**

out-of-gamut (OG) pixels. Table 5.1 shows the results on the 1,000 test images in full resolution (around 3000×5000). From the tables, it is clear that the best performance is obtained from metadata-assisted approach.

5.4.3 Qualitative results

We show qualitative visual output of some approaches. As with the previous chapter, figures with qualitative results are provided at the end of the chapter. For each method, the first row shows the predicted ProPhoto image. The second row shows an error map of the RMSE of each pixel between predicted and original ProPhoto. The third row shows the out-of-gamut colors after restoration on a CIE-xy chromaticity diagram. Figure 5.2, 5.3, and 5.4 show visual results of wide-gamut restoration between Photoshop, our GamutNet and our ProPhoto sampled local mapping method. Similar to the quantitative results, we see our approach achieves better results in terms of RMSE. In addition, the CIE-xy chromaticity diagram shows the recovered colors appear more like the ground truth wide-gamut image.

5.5 Summary

In summary, we have proposed our second method for wide-gamut restoration that works by strategically embedding metadata of original wide-gamut values within the small-gamut sRGB image as specialized metadata during a gamut reduction stage. This technique leverages the standard process within camera systems where sensor-RGB images are initially converted to the wide-gamut ProPhoto color space. Leveraging these samples, we outlined the creation of both global and local mapping functions, designed to reconstruct the clipped color values effectively. Our second method can address and resolve the generalization challenge encountered by GamutNet, as discussed in Chapter 4, through the

use of original wide-gamut value metadata for enhanced gamut restoration. Furthermore, we conducted a evaluation of our method, comparing its performance against both traditional and deep-learning-based methods, including the GamutNet approach discussed in Chapter 4. This comparison highlights our second method's efficacy in enhancing color restoration practices.

Our second strategy presents a limitation due to its requirement to integrate samples from the original wide-gamut colors into the sRGB output. When dealing with larger images or those containing a diverse array of colors outside the sRGB gamut, enhancing restoration quality to ProPhoto RGB means increasing the quantity of wide-gamut samples. This escalation in sample size directly leads to a larger metadata footprint within the image file. Moreover, our approach to local color mapping, while effective, can require up to 45 seconds of processing time on a CPU. This processing delay may render the method less practical for scenarios where instant results are essential.

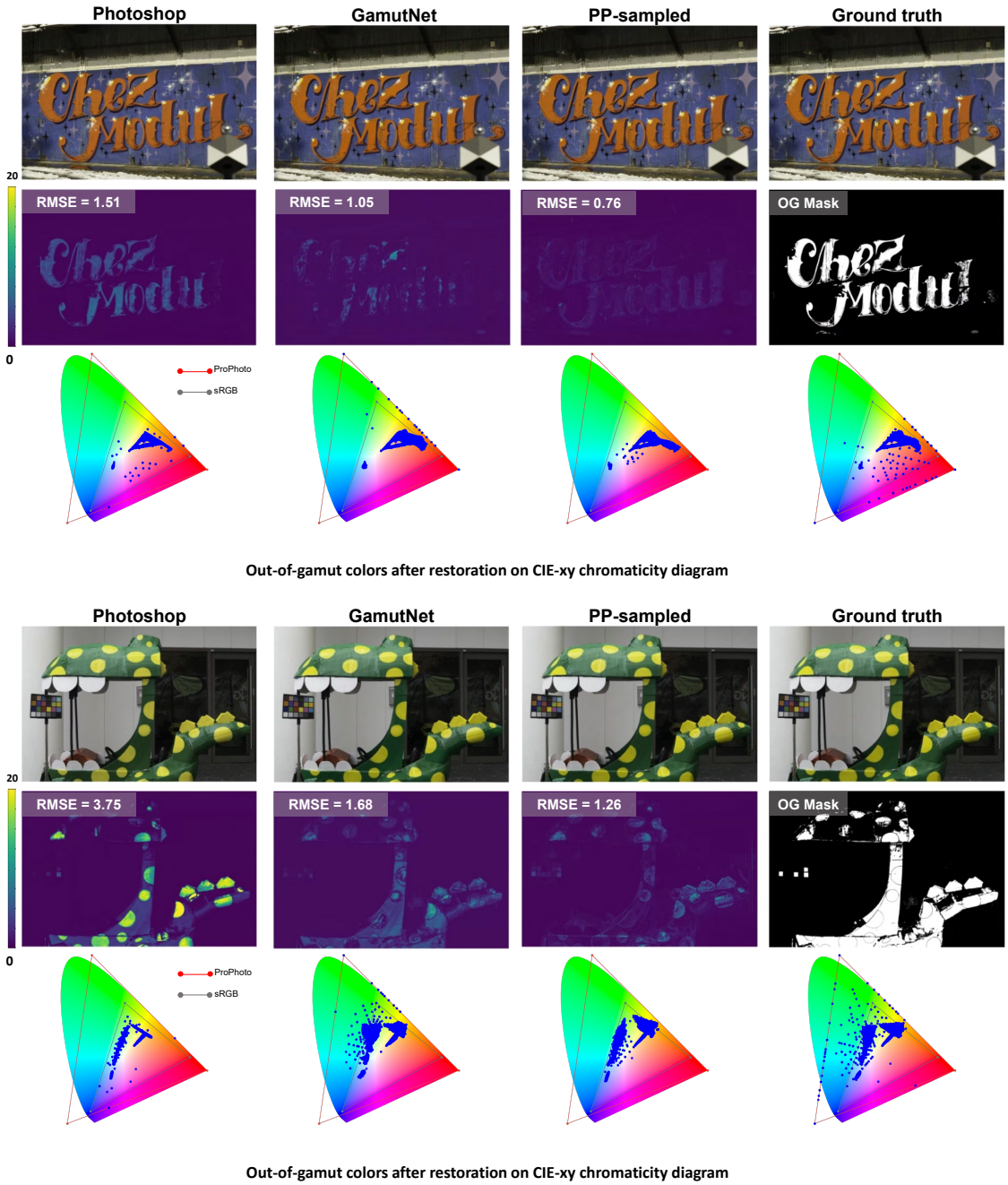
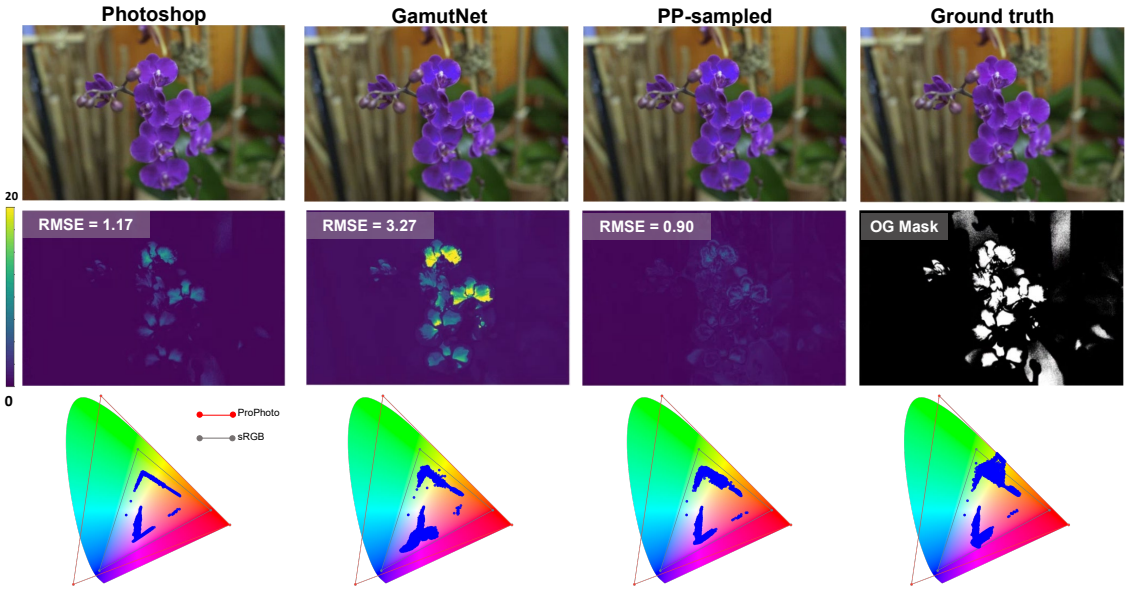


Figure 5.2: Comparisons among the ProPhoto results of Photoshop [40], our GamutNet, and our ProPhoto-sampled local mapping method. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto.



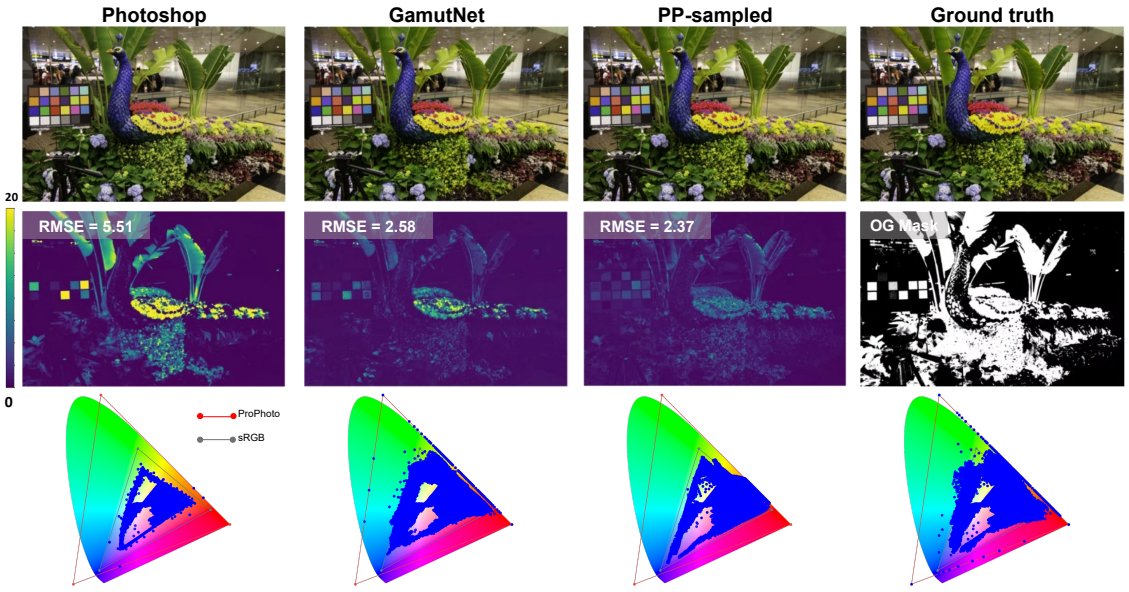


Figure 5.3: Comparisons among the ProPhoto results of Photoshop [40], our GamutNet, and our ProPhoto-sampled local mapping method. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto.

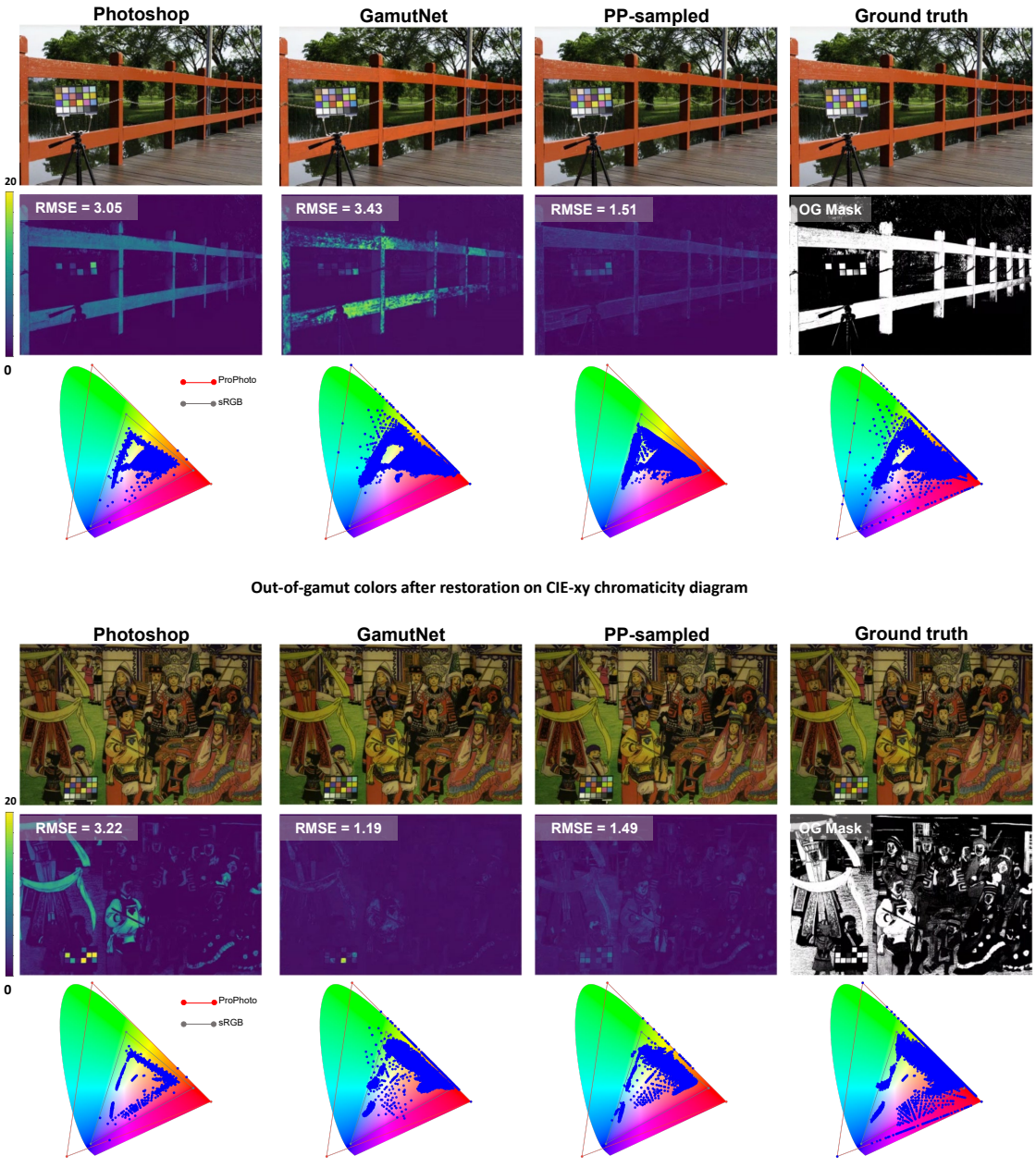


Figure 5.4: Comparisons among the ProPhoto results of Photoshop [40], our GamutNet, and our ProPhoto-sampled local mapping method. Heat maps of per-pixel RMSE are shown as well as plots of out-of-gamut colors on a CIE xy-chromaticity diagram showing the gamut for sRGB and ProPhoto.

6 Gamut Restoration via MLP

In this chapter, we describe our final approach for recovering wide-gamut colors that use a lightweight MLP. Finally, we explain our experiments and results. Our code and dataset can be found on the project website: `https://gamut-mlp.github.io`.

6.1 Introduction

When converting between color spaces, it is essential to address the gamut mismatch. Various strategies are employed for both gamut reduction and gamut expansion to achieve optimal results in each case. The most common approach for both stages uses absolute colorimetric strategy (clipping or hard clipping), where the goal is to minimize color distortion between the two gamuts. For example, in the case of gamut reduction from ProPhoto to sRGB, colorimetric errors are minimized by projecting (and clipping) out-of-gamut (OG) ProPhoto color values to the boundary of the sRGB space. When converting back from sRGB to ProPhoto, the absolute colorimetric strategy minimizes color error by leaving the clipped values untouched. Absolute colorimetric reduction and expansion is a common strategy used by consumer cameras eand image-editing software, such as Adobe Lightroom, DarkTable, and RawTherapee. Correcting the loss of color fidelity for color expansion is the goal of this work. A less common approach for reduction and expansion is to use soft-clipping [72], where the out-of-gamut values in ProPhoto are compressed to

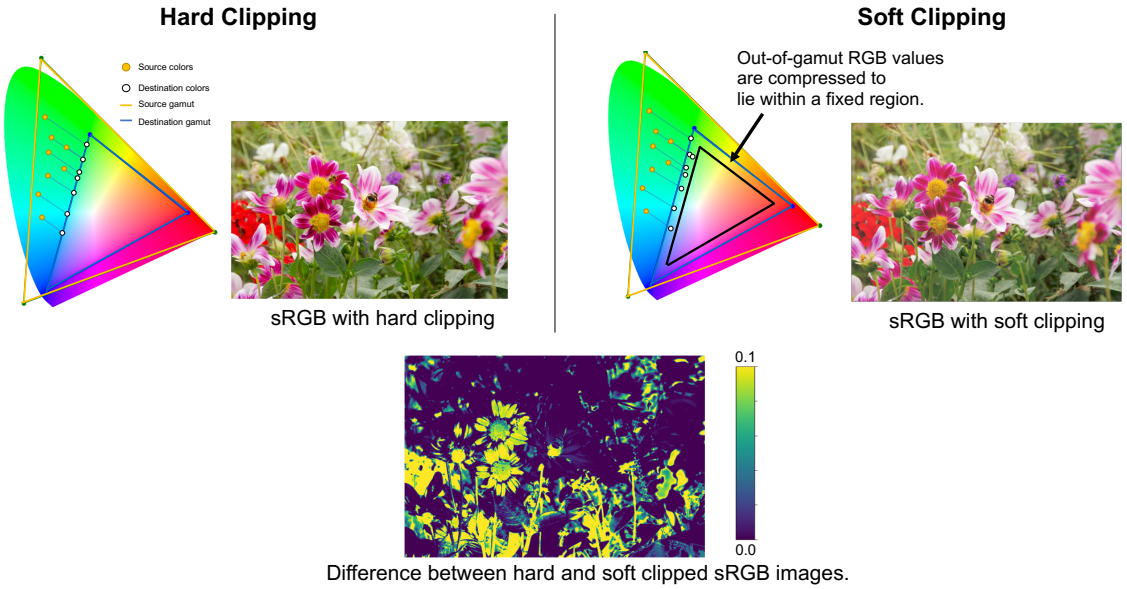


Figure 6.1: A comparison between clipping (hard clipping) and soft clipping.

fit within a specified region in the sRGB gamut. For example, instead of clipping out-of-gamut ProPhoto values, they are compressed to fit within the outer 10% of the sRGB gamut. On gamut expansion, the compressed region is expanded back to fill the ProPhoto gamut. While soft-clipping helps restore wide-gamut color values, it incurs colorimetric error during the gamut reduction to sRGB; as a result, most cameras and software do not use this. Figure 6.1 illustrates the difference between clipping (hard clipping) and soft clipping.

Building on the idea of embedding specialized metadata into small-gamut images during the gamut reduction process, as discussed in the previous chapter, we introduce our third strategy for both gamut reduction and gamut expansion. This approach also utilizes standard procedures within camera systems. Rather than embedding a limited amount of original wide-gamut values during the reduction stage and then creating color space mapping functions using these samples during the expansion stage, we propose optimizing a lightweight MLP to learn the mapping functions and embedding the MLP's parameters

into small-gamut images during the reduction step. This optimized MLP can then be reconstructed with those parameters and used in the expansion stage, thereby improving efficiency. Note that in this framework, the output images from gamut reduction are still the same as the ones from the clipping procedure, similar to our previous methods. As a result, the output images from our reduction do not have much color distortion.

In this chapter, we address the problem of recovering the RGB colors in sRGB images back to their original wide-gamut RGB representation by using a lightweight MLP. Our work is inspired by coordinate-based implicit neural image representations that use multi-layer perceptrons (MLPs) as a differentiable image representation. We propose to optimize a lightweight (23 KB) MLP model that takes the gamut-reduced RGB values and their spatial coordinates as input and predicts the original wide-gamut RGB values. The idea is to optimize the MLP model when the ProPhoto image is saved to sRGB and embed the MLP model parameters in the sRGB image as a comment field. The lightweight MLP model is extracted and used to recover the wide-gamut color values when needed. We describe an optimization process for the MLP that requires $\sim$2 seconds per full-sized image. We demonstrate the effectiveness of our method against several different approaches, including other neural image representations and pre-trained deep-learning-based models. As part of this work, we have created a dataset of 2200 wide-gamut/small-gamut image pairs for training and testing.

6.2 GamutMLP for Color Recovery

We begin with a high-level overview of our gamut recovery framework in Section 6.2.1. Details on the MLP architecture and optimization are provided in Section 6.2.2.

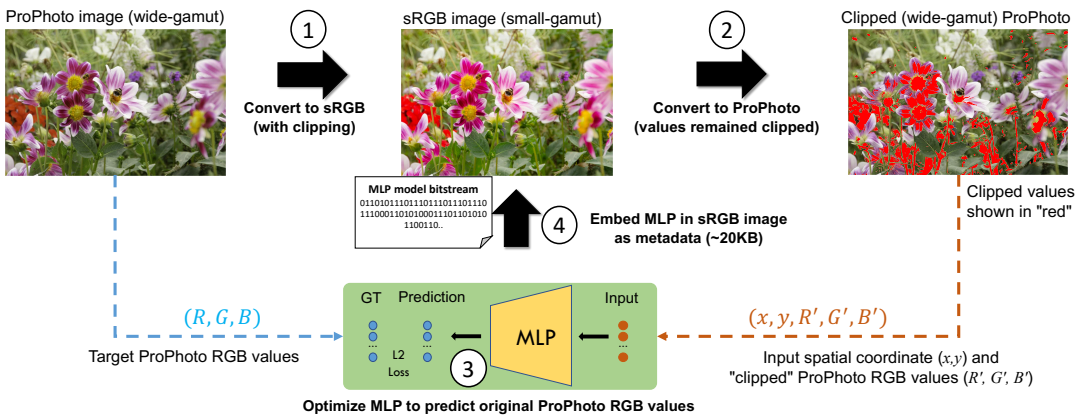


Figure 6.2: An overview of the *gamut reduction* stage in our framework. This phase shows the gamut reduction step, where the wide-gamut ProPhoto is converted to the small-gamut sRGB. While saving the sRGB image, an MLP is optimized based on the original and clipped ProPhoto color values. The MLP is embedded in the sRGB image as metadata.

6.2.1 Framework overview

Figure 6.2 and Figure 6.3 show illustrations of our framework's two steps: gamut reduction and gamut expansion.

Gamut reduction

The gamut reduction step is performed when the image is being converted from ProPhoto to sRGB, either on a camera or image editing software. We assume the input to be a wide-gamut ProPhoto RGB image denoted as $\mathbf{I}_{\text{PP}} \in \mathbb{R}^{3\times N}$, where N is the number of pixels. Gamut reduction is performed using the absolute colorimetric intent described in the previous section. The original ProPhoto image is transformed to the unclipped sRGB

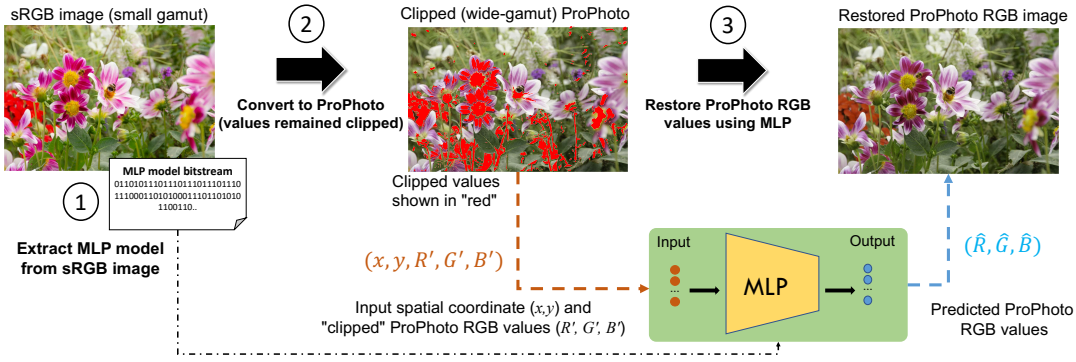


Figure 6.3: This figure provides an overview of the *gamut expansion* phase in our framework. In particular, the MLP network is extracted and used to recover the clipped ProPhoto values.

image using a 3×3 matrix such that the in-gamut sRGB values fall within the range $[0, 1]$. Out-of-gamut sRGB values are then clipped and processed with a gamma encoding to produce the final sRGB image. This procedure can be written as:

$$\mathbf{I}_{\text{sRGB}} = g(\text{clip}(\mathbf{M}\mathbf{I}_{\text{PP}}, min = 0, max = 1)), \tag{6.1}$$

where $\mathbf{M}$ is the matrix that maps between ProPhoto and the unclipped sRGB, clip() is the clipping operation, and g is the gamma-encoding for sRGB [22].

When converting sRGB colors back to ProPhoto RGB using the inverse transforms, the clipped color values will not be recovered. We refer to this clipped ProPhoto image as $\mathbf{I}_{\text{ClippedPP}} \in \mathbb{R}^{3\times N}$. Pixels with clipped values are illustrated in red in Figure 6.2. We express the mapping from sRGB to clipped ProPhoto as:

$$\mathbf{I}_{\text{ClippedPP}} = \mathbf{M}^{-1} g^{-1}(\mathbf{I}_{\text{sRGB}}), \tag{6.2}$$

where $g^{-1}(\cdot)$ is a de-gamma function for the input sRGB image, and $\mathbf{M}^{-1}$ is the inverse transform to convert the sRGB image back to the ProPhoto color space.

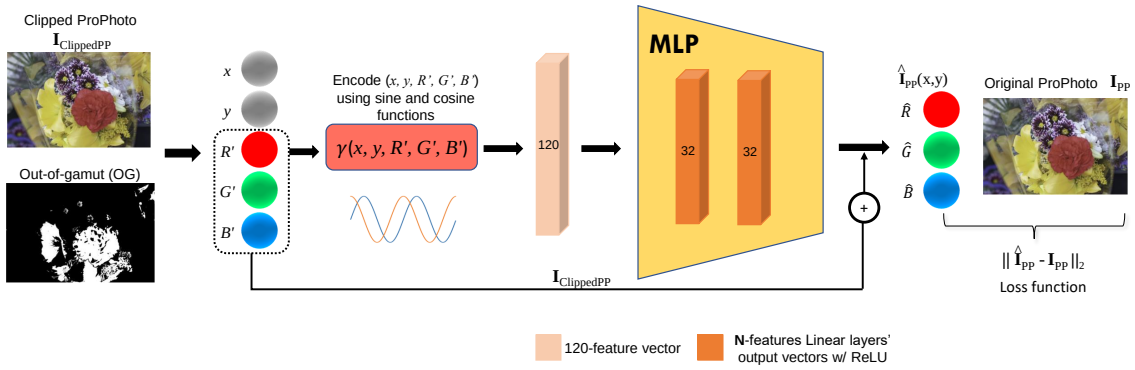


Figure 6.4: This figure shows the GamutMLP architecture. Given the clipped ProPhoto image, we optimize the MLP using samples from in-gamut and out-of-gamut pixels. The 5D coordinate and color input (x, y, R', G', B') is encoded as a 120D-feature vector before passing it to the MLP. The MLP has three linear layers, with the final layer predicting a residual to add to the R', G', B' input. The loss is computed against the original ProPhoto image R, G, B values.

Applying Eqs. 6.1 and 6.2, we have the original ProPhoto image, $\mathbf{I}_{\text{PP}}$, and its clipped $\mathbf{I}_{\text{ClippedPP}}$. We also know which values were clipped. Using these two images, we optimize a lightweight MLP (GamutMLP) to predict a residual value that, when added to the $\mathbf{I}_{\text{ClippedPP}}$ recovers $\mathbf{I}_{\text{PP}}$. The GamutMLP model parameters are embedded in the sRGB image when it is saved. Since the parameters of our MLP require only 23 KB of memory, this can easily be embedded as a comment field in the image.

Gamut expansion

Given an sRGB image with an embedded GamutMLP model, we extract the model and perform the standard color space conversion described in Eq. 6.2 to compute $\mathbf{I}_{\text{ClippedPP}}$. The extracted model predicts the residuals of all pixels and adds them to the $\mathbf{I}_{\text{ClippedPP}}$ to

recover the color values as shown in Figure 6.3.

The following section provides details of the GamutMLP model architecture and its optimization.

6.2.2 GamutMLP model and optimization

GamutMLP architecture

Figure 6.4 shows a diagram of the GamutMLP architecture. When converting from ProPhoto to sRGB, we keep track of the transformed RGB values that lie outside the sRGB gamut. These pixel locations are denoted in the out-of-gamut mask in Figure 6.4. Out-of-gamut pixels will be clipped to fit within the sRGB gamut.

The MLP input is a 5D vector of a pixel's spatial coordinates (x, y) and color value (R', B', G') from the clipped wide-gamut ProPhoto image. GamutMLP predicts the residual that needs to be added to $\mathbf{I}_{\text{ClippedPP}}$ to recover the wide-gamut original (R, G, B). We can express GamutMLP as follows:

$$\hat{\mathbf{I}}_{\text{PP}}(\mathbf{x}) = f_\theta(\mathbf{x}, \mathbf{I}_{\text{ClippedPP}}(\mathbf{x})) + \mathbf{I}_{\text{ClippedPP}}(\mathbf{x}), \tag{6.3}$$

where f_θ represents the *GamutMLP*, θ is the model's parameters, and $\hat{\mathbf{I}}_{\text{PP}}$ is the final recovered ProPhoto image. The MLP's input values (x, y, R', G', B') are normalized to the range $[-1, 1]$, and then pass to the encoding function γ. The use of an encoding function has been shown effective in improving neural implicit representations optimization [68,95]. Prior neural implicit representations tend to have only 2D coordinates as input, while we apply the encoding function to both spatial coordinates and RGB values. We found the following mapping worked well for our task:

$$\gamma(m) = (\sin(2^0\pi m), \cos(2^0\pi m), ..., \sin(2^{K-1}\pi m), \cos(2^{K-1}\pi m)), \tag{6.4}$$

where m is a spatial coordinate or RGB value. In our experiment, we choose K = 12. The γ function projects each of the 5D input values to a 24-dimension encoding, resulting in a final 120D feature vector for each input. The GamutMLP has three linear layers. The first two are fully connected ReLU layers with 32 output features. The last layer outputs three values and has no activation function. Our MLP is optimized with an L_2 loss function computed between the predicted ProPhoto image and the original ProPhoto image:

$$\mathcal{L}_{gamut} = \sum_{\mathbf{x}} ||(\hat{\mathbf{I}}_{\text{PP}}(\mathbf{x}) - \mathbf{I}_{\text{PP}}(\mathbf{x}))||_2^2. \tag{6.5}$$

Pixel sampling and standard optimization

We describe two optimization strategies: standard and fast. For our standard optimization, model parameters are randomly initialized. At optimization time, we know which pixels are out-of-gamut (OG) and which are in-gamut (IG). We found that training the MLP on both out-of-gamut pixels and in-gamut (non-clipped) pixels gave the best result. We optimized our model by sampling 2% of the IG pixels and 20% of the OG pixels uniformly over their spatial coordinates. For all reported results, the model was optimized for 9,000 iterations with a learning rate of $1e-3$ using the Adam optimizer [52].

Faster MLP optimization

To speed up our optimization, we used the recent methods proposed by [25,76] to improve optimization time. We also incorporated a meta-learning strategy [84] to pre-train a generic GamutMLP whose model parameters can be used for initialization. Such initialization has been shown to help coordinate-based MLPs converge faster during optimization [104].

To pre-train a meta-MLP model, we used images from our training dataset described in the following section. For each meta epoch, the meta-MLP fits each image with 10,000 iterations using a larger learning rate $1e-2$ and the SGD optimizer. When a per-image

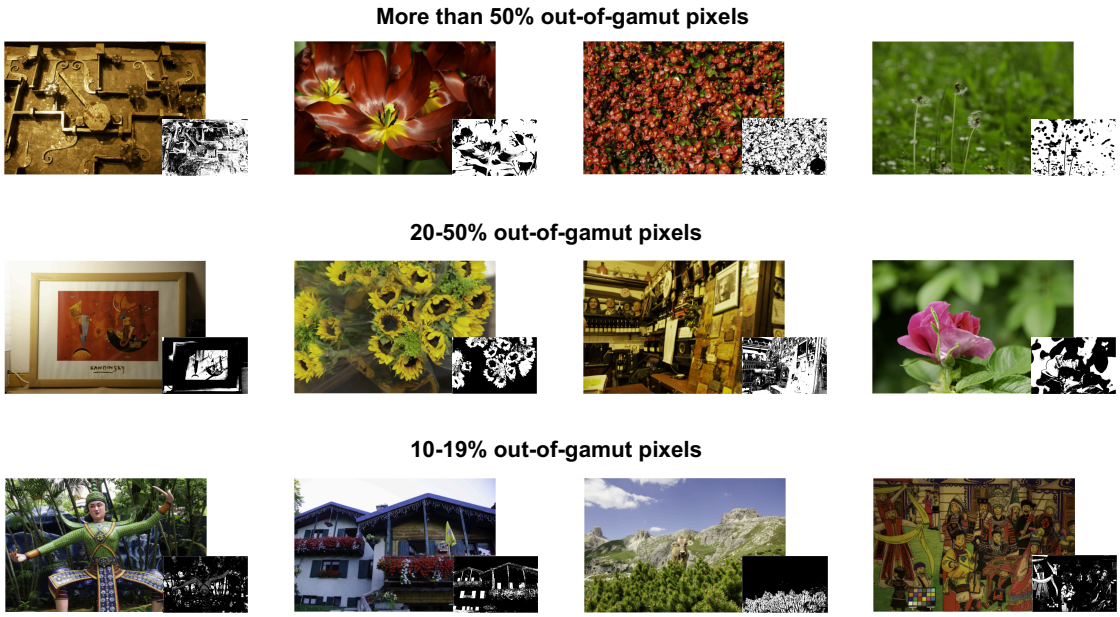


Figure 6.5: Examples from our dataset showing different amounts of out-of-gamut pixels (OG pixels are white in the masks).

GamutMLP is initialized with this pre-trained MLP, our optimization requires only 1,200 iterations instead of 9,000. Combining our meta-MLP for initialization with [25, 76] significantly reduced optimization time.

6.3 Dataset and Results

We first describe our dataset generation for evaluating this work. The images in our dataset are employed in training our meta-GamutMLP model (for weight initialization) as well as other competing DNN-based methods, and for assessing outcomes.

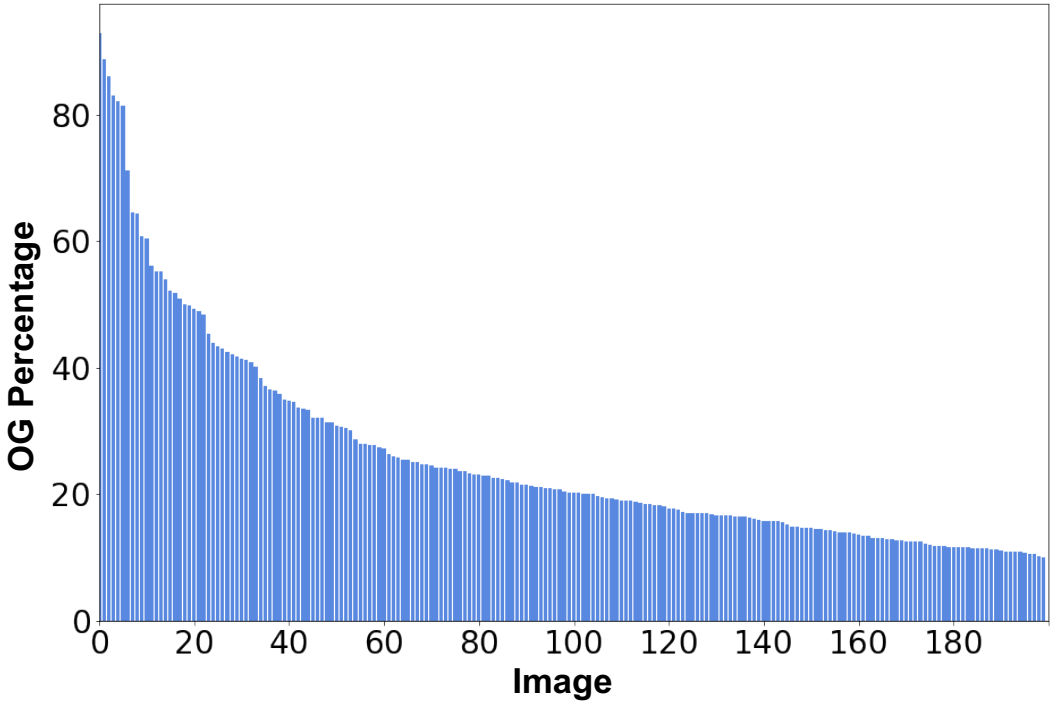


Figure 6.6: A histogram of our dataset in terms of percentage of out-of-gamut pixels.

6.3.1 Dataset

In this chapter, we aim to create a dataset that is more diverse for our experiments than the one described in Chapter 4. To prepare wide-gamut ProPhoto images, we followed a procedure used in 4 that processed RAW images from the MIT-Adobe FiveK [16], RAISE [23], Cube+ [11], and NUS [21] public datasets.

There are 16,599 RAW images from these four datasets, representing a wide range of scene content. Figure 6.7 presents a selection of RAW images we collected, encompassing a variety of scenes both indoors and outdoors. The collection features diverse types of subjects such as humans, flowers, cars, landscapes, and more. We use Adobe Camera RAW (ACR) to mimic a camera ISP to render the RAW images to 16-bit wide-gamut

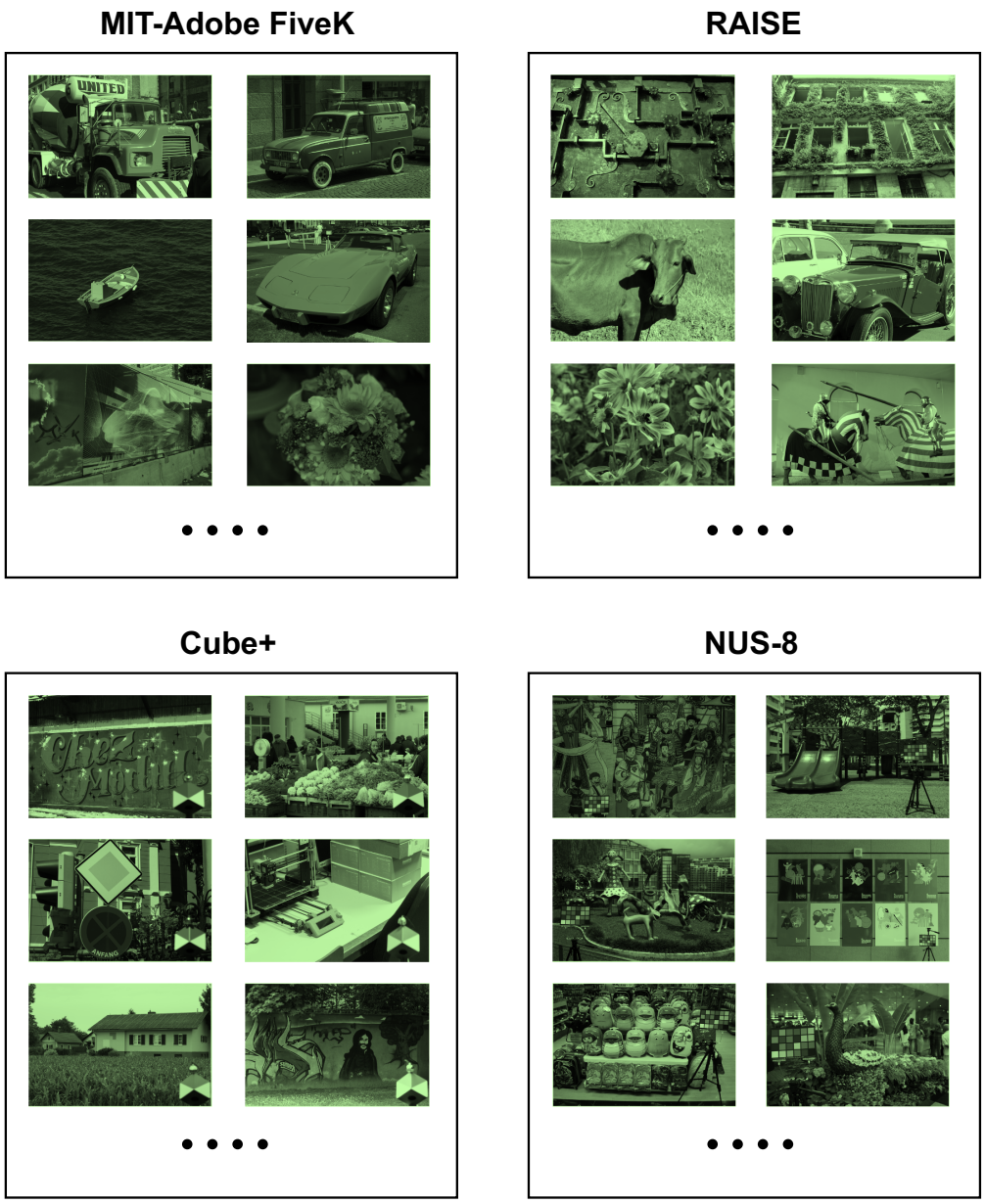


Figure 6.7: RAW images collected from four different sources: MIT-Adobe FiveK [16], RAISE [23], Cube+ [11], and NUS [21].

ProPhoto images. ACR can apply different photo-finishing styles when rendering RAW images. We use four picture styles—Adobe Standard, Adobe Landscape, Adobe Color,

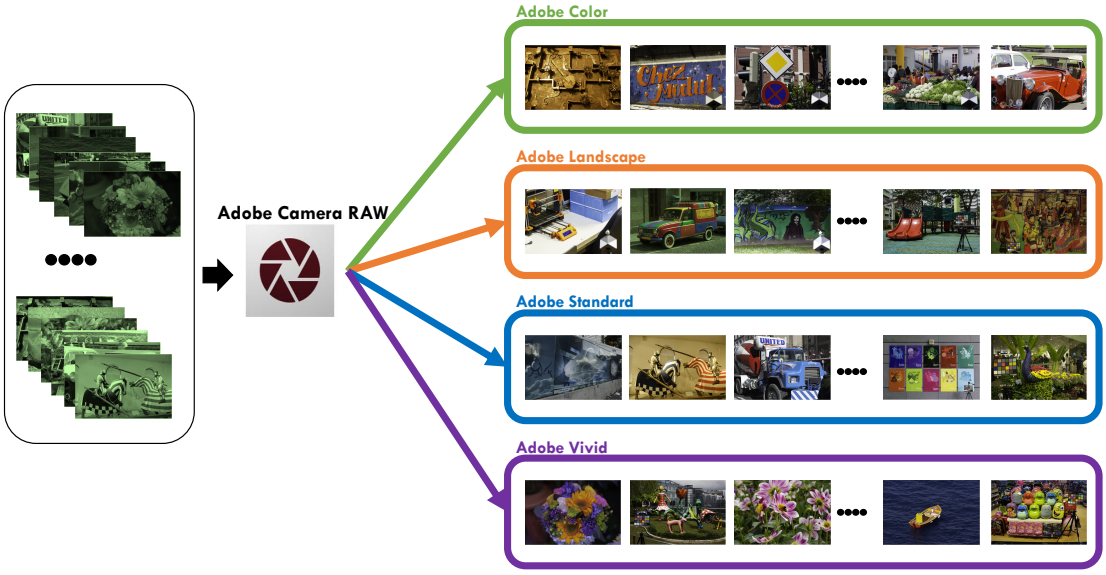


Figure 6.8: RAW images rendered to ProPhoto using four different picture styles: Adobe Standard, Adobe Landscape, Adobe Color, and Adobe Vivid.

and Adobe Vivid—to generate ProPhoto images with vivid colors, expanding on the two styles employed in 4. Figure 6.8 illustrates this process. These are saved in their original full resolution size, ranging from 2000×3000 to 4000×6000.

From these rendered images, we chose 2000 images for the training set and 200 for the testing set. Images were selected such that the percentage of out-of-gamut pixels is between 10% and 90% (the dataset in Chapter 4 contains only up to 50% of out-of-gamut colors). Figure 6.5 shows samples from our dataset illustrate varying levels of out-of-gamut pixels, with these pixels displayed as white in the masks. Also, Figure 6.6 displays a histogram that represents the percentage of out-of-gamut pixels in our dataset. The minimum number of out-of-gamut pixels for an image in our testing set is over a million pixels. The final breakdown of images selected from the starting datasets is: 11% Adobe FiveK, 62% RAISE, 12% Cube+, and 15% NUS.

6.3.2 Comparisons

The following describes methods used for comparisons against the GamutMLP approach.

Conventional methods

The two baseline conventional methods are clipping and soft-clipping [72], described in Section 6.1. Clipping is currently the de facto method for gamut reduction. While soft-clipping aids in gamut expansion, it is not commonly used because it distorts the colors in the sRGB.

Pre-trained deep networks

For pre-trained DNN, we first compare with our first approach GamutNet, which is a DNN-based method targeting gamut recovery [56], as described in Chapter 4. For the sake of completeness, we also compare against several image-to-image-translation methods: pix2pix [42], pix2pixHD [107], and ASAPNet [99]. We can consider our problem of clipped-ProPhoto to ProPhoto conversion as a special application for image-to-image translation methods. For all methods that need to be pre-trained, we train them with 512×512 crops from training images (clipped ProPhoto and ProPhoto pairs). Cropping is performed such that at least 10% of the cropped image has out-of-gamut pixels. At inference time, the DNN-based methods are applied to the full-sized testing images.

Per-image optimization

We also compare with [55], another metadata approach that uniformly samples pixels (135KB) of the original ProPhoto image for recovery using polynomial color correction functions. We also compare with several variants of the SIREN [100] coordinate-based neural implicit function. In particular, we start with the original SIREN, which uses 2D

coordinates for input, and has five fully connected linear layers, each with 256 channels and periodic activation functions; the last linear output layer has only three channels for RGB values. The SIREN model requires 796 KB. We optimize a SIREN-residual variant that predicts the residual between the clipped and ground-truth ProPhoto images. Finally, we try a variant of SIREN-residual that is limited to a small model size (69 KB) to mimic a smaller model. The SIREN models are optimized based on the loss $\mathcal{L}_{gamut}$ described in Equation 6.5. For the other hyperparameters, we adopt default settings for image-fitting tasks from SIREN [100]. Unfortunately, we could not use the fast implementation of MLP [25, 76] for the original SIREN and variants since the fast implementation API does not support the sinusoidal activation function used by SIREN.

For our GamutMLP approach, we show the results of our MLP variants with encoded inputs (i.e., Equation 6.4), and with and without optimization plus the meta-GamutMLP initialization. The pre-trained DNNs methods and MLP-based approaches are trained or optimized using a NVIDIA Quadro RTX 6000.

6.3.3 Quantitative results

Table 6.1 summarizes the performance of the tested methods. Results are reported as the average root mean square error (RMSE) and peak signal-to-noise ratio (PSNR) between the predicted and ground truth test images. Metrics are provided for the entire image and for only out-of-gamut (OG) pixels. We provide the associated metadata size (KB) and optimization times for the relevant methods.

The table reveals that simple image-to-image translation does not work well for this task. Pix2pix [42] gives better results than Pix2pixHD [107], since Pix2pix is a general-purpose method for image-to-image translation, while Pix2pixHD was designed to synthesize images from semantic label maps. The DNN-based GamutNet [56] method explicitly

designed for gamut recovery performs better than the clipping baseline. The best results are obtained from the per-image optimized methods. In particular, the full-sized SIREN-residual MLP model produces results similar to ours. However, the model requires 796KB and is slow to optimize even compared with our equivalent standard optimized MLP. Our GamutMLP with feature encoding and pre-trained initialization gives the best results and the fastest optimization performance. Note that the ProPhoto-sampled method [55] is optimized per image; however, the computationally intensive component is during the gamut expansion phase instead of the reduction phase. In addition, the method's implementation uses Matlab and runs on CPU, so we do not include its optimization time in our table since the comparison is unfair.

6.3.4 Training details for baseline methods and inference time

The four baseline DNNs used different training sizes in their github repositories. We felt to be fair we should use a common dataset when training all methods. The 512×512 crop was the most common input size. The per-image optimization approaches were all significantly better than pre-trained DNNs, so small improvements based on different training sizes might not matter.

Inference time over the test dataset of our lightweight MLPs is around **0.27 seconds**, while it takes DNN-based methods approximate **0.57 seconds** in average with GPU. The ProPhoto-Sampled Local Mapping that uses a spatially varying mapping functions take up to 45 seconds per image on CPU [55].

We note that the MLP-based methods purposely over-fit the MLP to the input image. Longer optimization gives better PSNR. However, to be practical we wanted the optimization to be within 2 seconds.

6.3.5 Qualitative results

We show qualitative visual outputs of selected approaches in Figure 6.9, Figure 6.10, Figure 6.11, Figure 6.12, Figure 6.13, and Figure 6.14. As done in previous chapters, all subjective results are provided at the end of the chapter. For each method, the first row shows the predicted ProPhoto image. The second row shows an error map of the RMSE of each pixel between the predicted and original ProPhoto. The third row shows the out-of-gamut colors after restoration on a CIE-xy chromaticity diagram. Similar to the quantitative results, we see our approach achieves better results in terms of RMSE and PSNR. In addition, the CIE-xy chromaticity diagram shows the recovered colors appear more like the ground truth wide-gamut images.

While not the goal of our work, on careful observation, it can be noticed that our method and the ProPhoto-sampled method [55] provide a slight improvement in the in-gamut pixels. This is attributed to a slight recovery of some of the 16-bit values that were quantized when the image was saved in 8-bit sRGB in the gamut reduction step. Recall that this method includes in-gamut pixels in the optimization and is applied by the GamutMLP to all pixels. Similarly, the work [55] uniformly samples ProPhoto pixels, including in-gamut and out-of-gamut pixels. The most significant improvements, however, are still with the clipped out-of-gamut pixels, which incur the most error in the gamut reduction step.

6.3.6 Ablations

We present some ablation studies conducted to determine the most effective hyper-parameters and the optimal design of our model architecture.

Input types

We trained our proposed MLP to explore the efficacy of various types of input data. These types included: coordinates only (represetned as x, y), color values only (defined by the RGB scale as R, G, B), and a comprehensive 5-dimensional vector that integrates both coordinates and color values (x, y, R, G, B), as detailed in Table 6.2. This experiemntal design allowed us to evaluate the impact of different types of input data on the MLP's performance. Remarkably, we discovered that leveraging both the spatial information provied by the coordinates and the visual information conveyed through the color values resulted in superior performance metrics, even with a relatively small neural network architecture. This finding underscores the significant advantage of combining these diverse data types, as it facilitates a more understanding and interpretation of the data by the model.

Encoding function

The feature encoding 6.4 used has been shown to be effective for implicit neural methods [68,95]. It makes a big difference in our task. See Table 6.3 below, where we show that our MLP with the encoding function is significantly better than the MLP without using the function.

GamutMLP model sizes

Our goal was to find a model size that was compact and had a reasonable optimization time. In particular, we tried to vary the hidden features from 128, 64, 32 to 16, and their corresponding model sizes are 137 KB, 53 KB, 23 KB, and 11 KB. Figure 6.16 shows a plot of the average PSNR computed over all these 200 test images for each model. Results are shown for the entire image and out-of-gamut pixels only. Figure 6.15 shows qualitative results on a test image for the different model sizes. While the larger model gave a slightly

better performance, the 23 KB model provided a comparable result, with memory size that can easily be included as a comment field in an sRGB image. The model below 23 KB performed poorly.

6.4 Summary

In this chapter, we have presented a framework to recover wide-gamut color values lost due to the gamut reduction step applied when converting a ProPhoto image to a sRGB image. We cast our task as a restoration problem with the goal of restoring the original color loss due to gamut clipping. By integrating our approach into the gamut reduction step when the image is converted to sRGB, we have the opportunity to optimize a model directly against the known restoration target—a luxury most restoration problems such as deblurring and image sampling do not have. This allowed us to use a lightweight MLP network to predict the original color signal. Compared to the overall sRGB image size (typically 2–5 MB), the 23 KB memory overhead (no matter the size of the image, even with dimensions up to 4000×6000, the storage space used is still the same) for the GamutMLP model is negligible and means that the color fidelity recovery is obtained virtually for free.

Our experiments show that per-image MLP optimization provides much better results than pre-trained DNN models for color recovery. Furthermore, our small GamutMLP provides comparable performance in PSNR compared with larger MLP-based neural implicit functions but requires significantly less memory size and optimization time (within 2 seconds). As part of this effort, we have also generated a new dataset of 2200 images with high-color fidelity that will be useful in advancing research in this area. Our code and dataset can be found on the project website: `https://gamut-mlp.github.io`.

Method	Metadata↓	RMSE↓	RMSE OG↓	PSNR↑	PSNR OG↑	Optim. Time↓
			Conventional			
Clip	-	0.0069	0.0126	43.22	37.98	-
Soft Clip	-	0.0039	0.0042	48.17	47.54	-
			Pre-trained DNN			
Pix2pix [42]	-	0.0087	0.0167	41.24	35.55	-
Pix2pixHD [107]	-	0.0157	0.0314	36.08	30.07	-
ASAPNet [99]	-	0.0518	0.0993	25.72	20.06	-
GamutNet [56]	-	0.0052	0.0088	45.75	41.08	-
			Optimized per image			
ProPhoto-Sampled [55]	135 KB	0.0032	0.0051	49.78	45.90	-
SIREN [100]	796 KB	0.0648	0.0421	23.77	27.52	115.67 mins
SIREN-residual	796 KB	0.0033	0.0044	49.72	47.20	118.98 mins
SIREN (small)-residual	69 KB	0.0040	0.0052	47.98	45.66	94.62 mins
MLP + enc. (no optimization)	48 KB	0.0021	0.0031	53.57	50.17	37.05 sec
MLP (53KB) + enc.	**53 KB**	**0.0021**	**0.0030**	**53.73**	**50.33**	**16.29 sec**
MLP (23 KB) + enc.	**23 KB**	**0.0021**	**0.0031**	**53.65**	**50.04**	**16.32 sec**
MLP (53KB) + enc. + meta init.	**53 KB**	**0.0021**	**0.0032**	**53.40**	**50.00**	**1.94 sec**
MLP (23 KB) + enc. + meta init.	**23 KB**	**0.0021**	**0.0032**	**53.36**	**49.93**	**1.90 sec**

Table 6.1: This table shows results on various methods used for wide-gamut color recovery. The reported numbers are the average results computed against the 200 16-bit ProPhoto ground-truth full-size images. RMSE and PSNR are provided for the whole image and out-of-gamut (OG) pixels. For the per-image methods, we provide associated metadata (size in KB) and optimization time. For pre-trained DNNs, we compare with Pix2pix [42], Pix2PixHD [107], ASAPNet [99], and GamutNet [56]. For the per-image methods, we compare with ProPhoto-Sampled (local mapping) [55], SIREN [100] variants, and our GamutMLP variants.

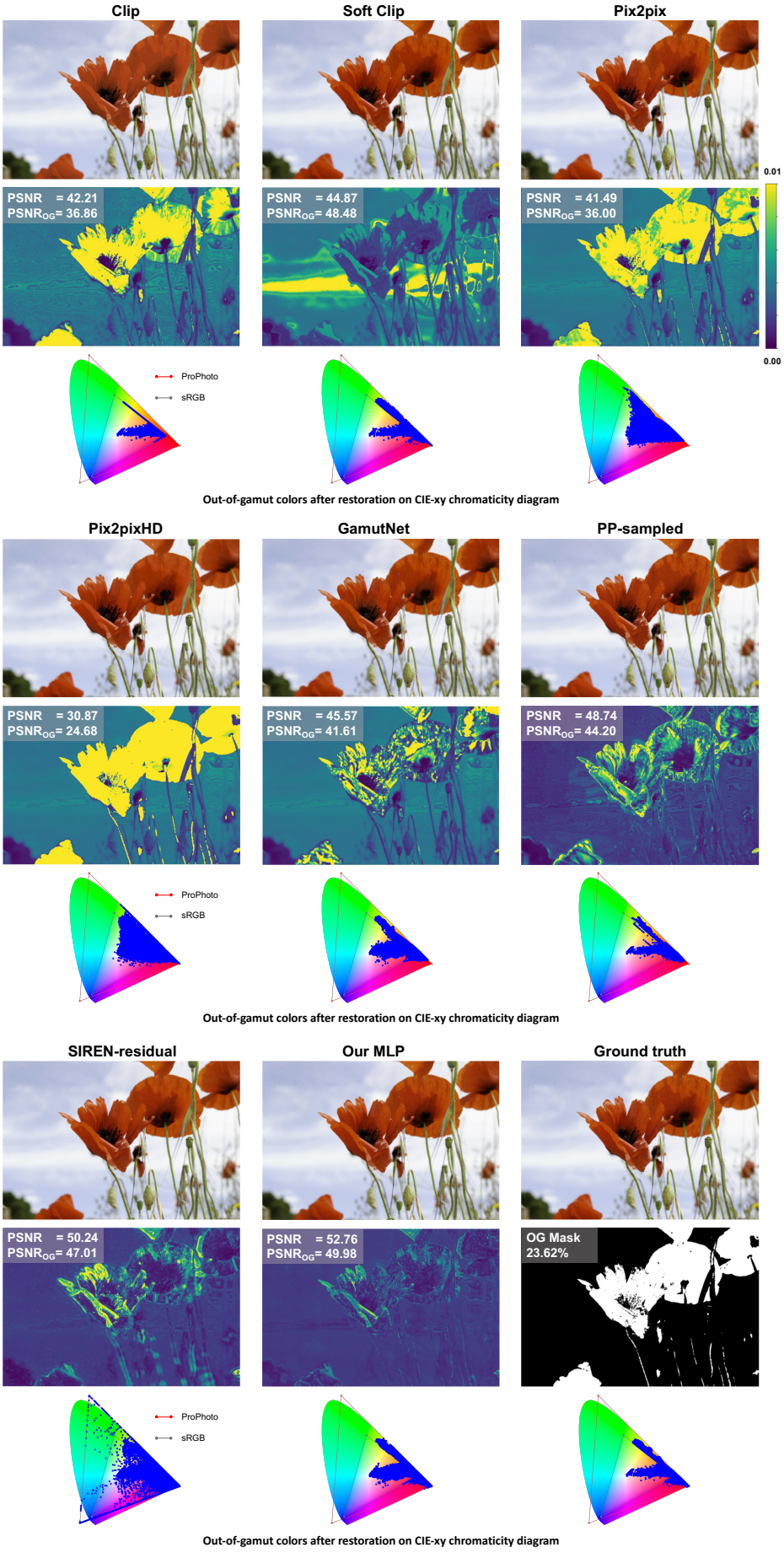


Figure 6.9: Qualitative comparisons between the predicted ProPhoto full-size output of various methods. Error maps of per-pixel RMSE and plots of out-of-gamut (OG) colors on CIE-xy chromaticity diagram with the gamuts of sRGB and ProPhoto are shown.

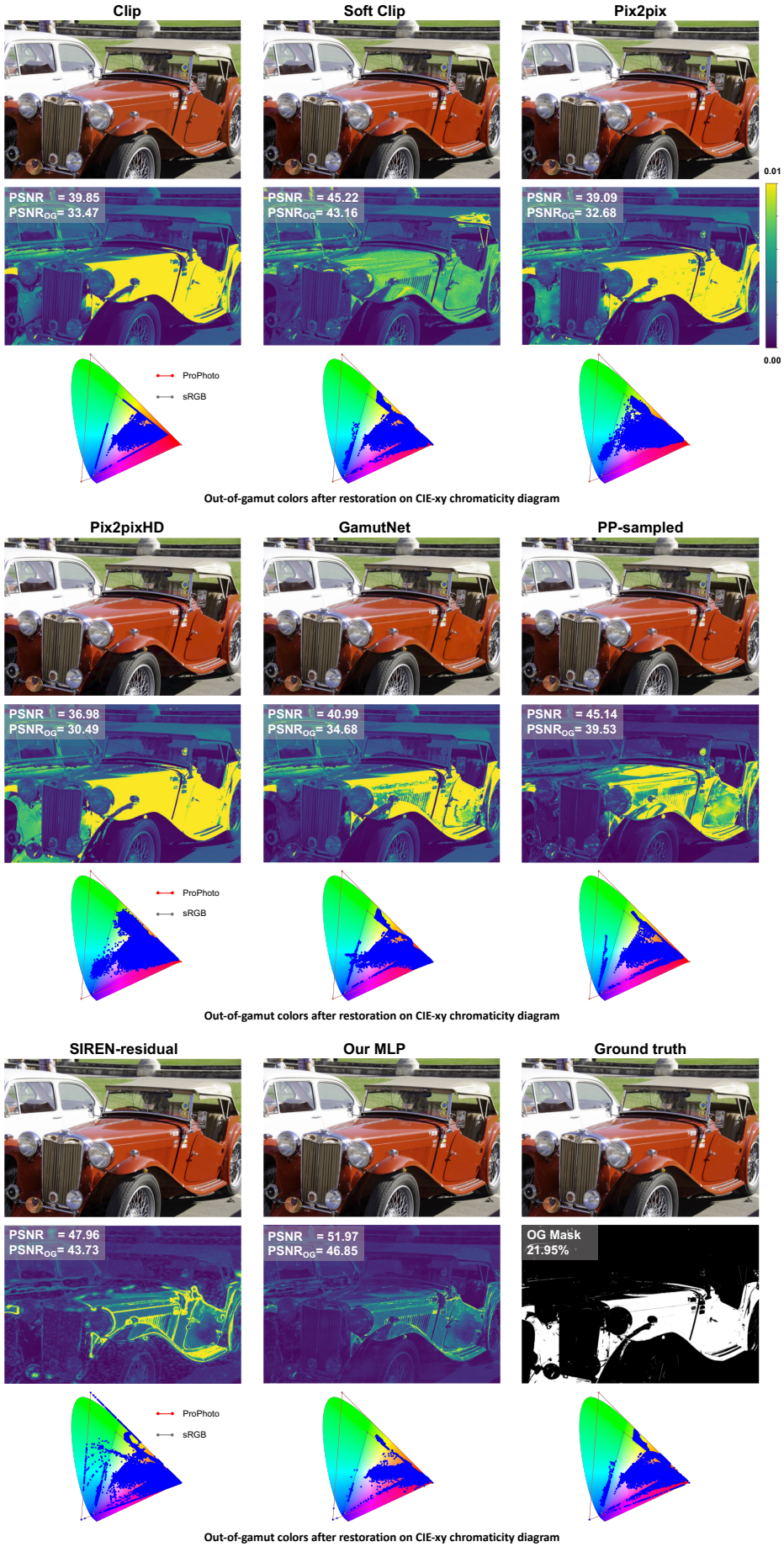


Figure 6.10: Qualitative comparisons between the predicted ProPhoto full-size output of various methods. Error maps of per-pixel RMSE and plots of out-of-gamut (OG) colors on CIE-xy chromaticity diagram with the gamuts of sRGB and ProPhoto are shown.

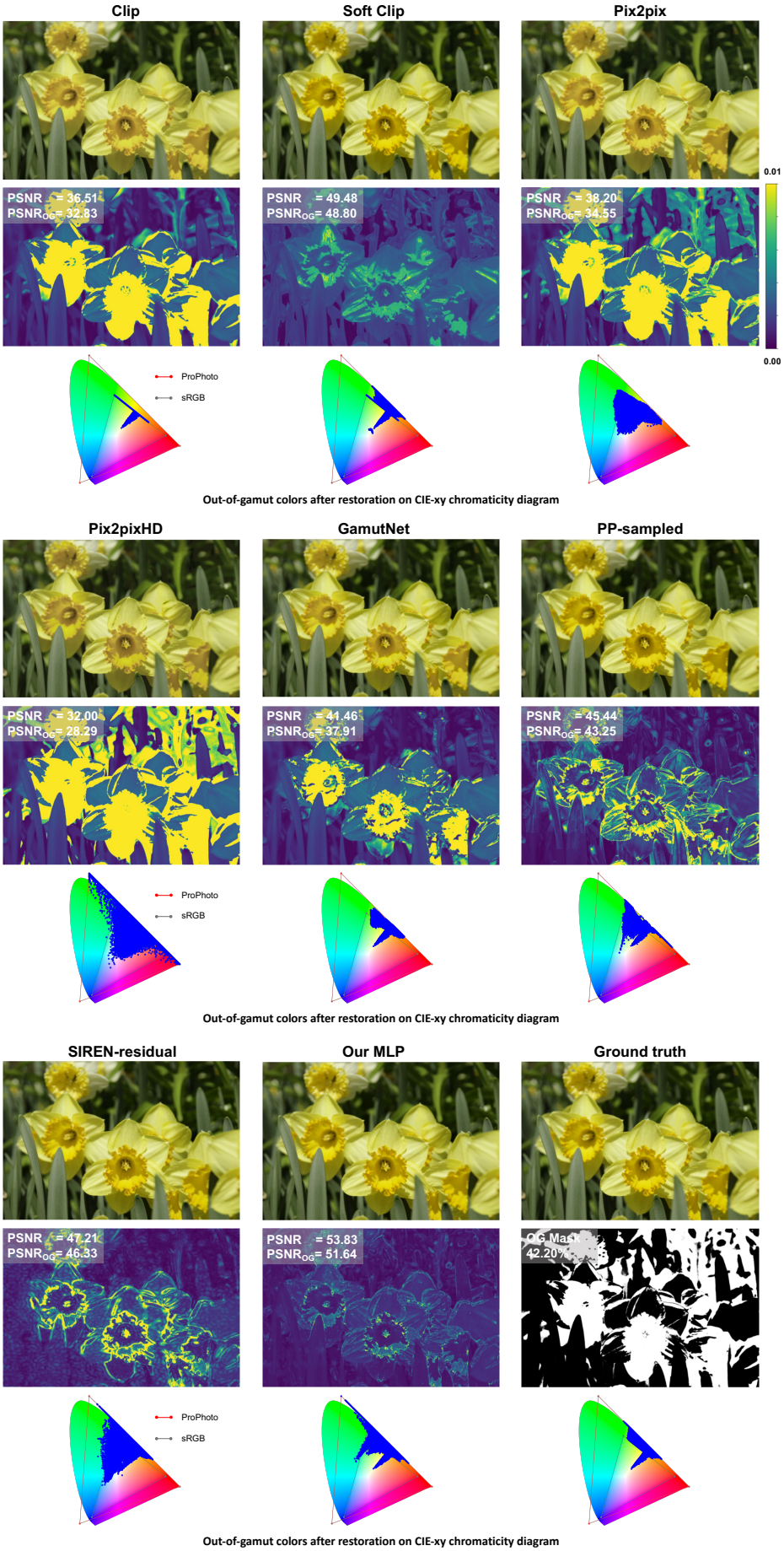


Figure 6.11: Qualitative comparisons between the predicted ProPhoto full-size output of various methods. Error maps of per-pixel RMSE and plots of out-of-gamut (OG) colors on CIE-xy chromaticity diagram with the gamuts of sRGB and ProPhoto are shown.

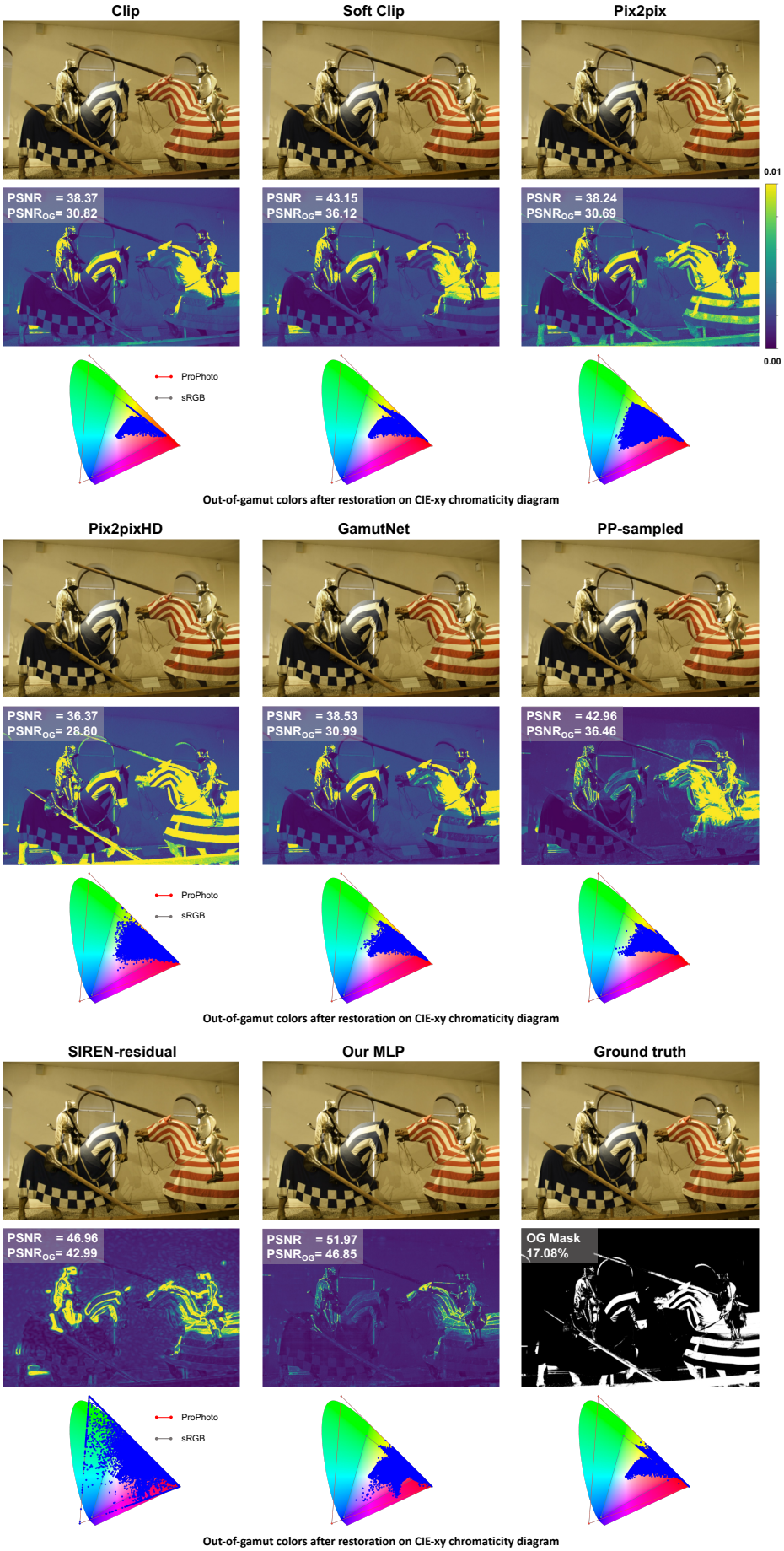


Figure 6.12: Qualitative comparisons between the predicted ProPhoto full-size output of various methods. Error maps of per-pixel RMSE and plots of out-of-gamut (OG) colors on CIE-xy chromaticity diagram with the gamuts of sRGB and ProPhoto are shown.

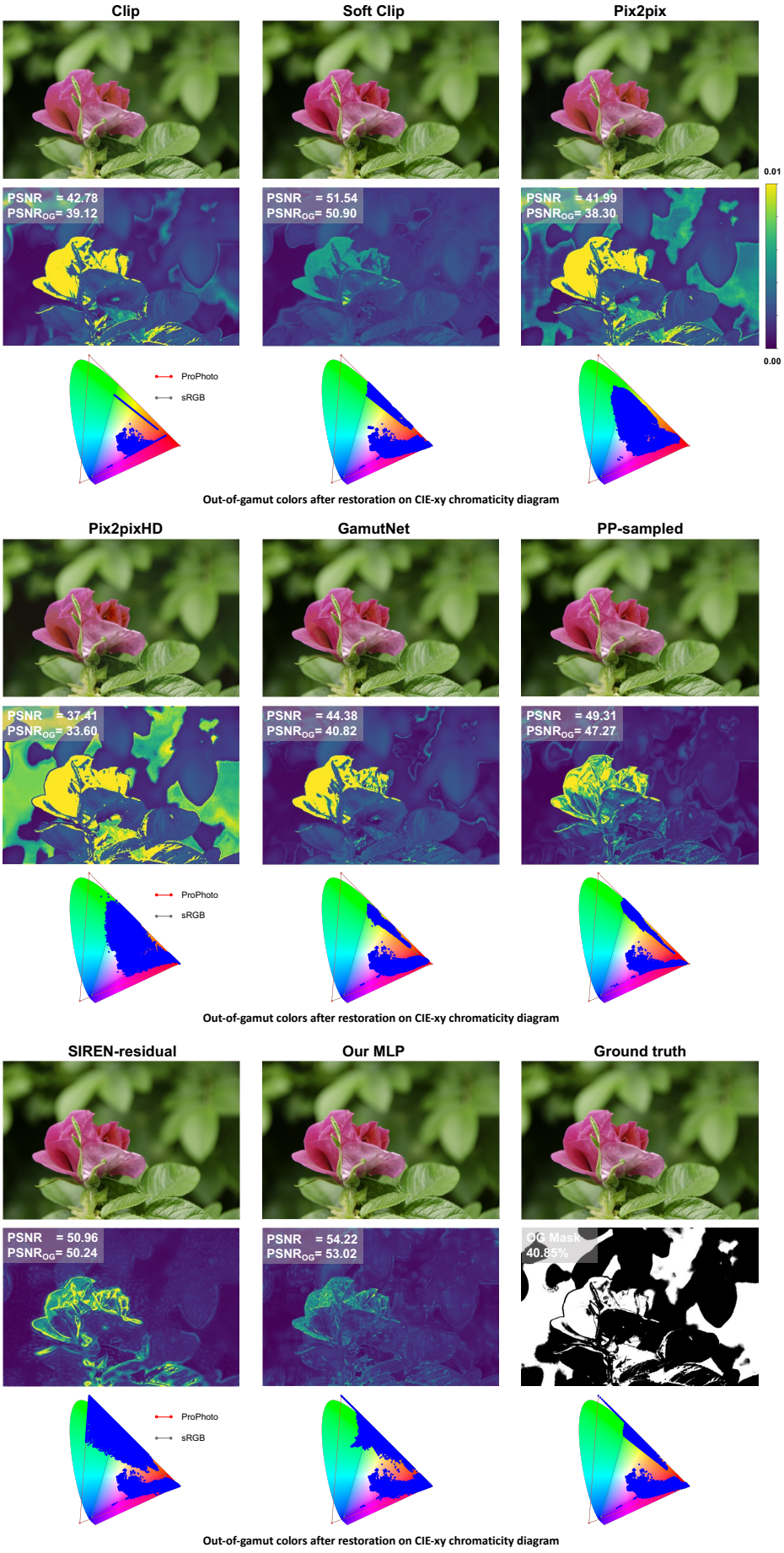


Figure 6.13: Qualitative comparisons between the predicted ProPhoto full-size output of various methods. Error maps of per-pixel RMSE and plots of out-of-gamut (OG) colors on CIE-xy chromaticity diagram with the gamuts of sRGB and ProPhoto are shown.

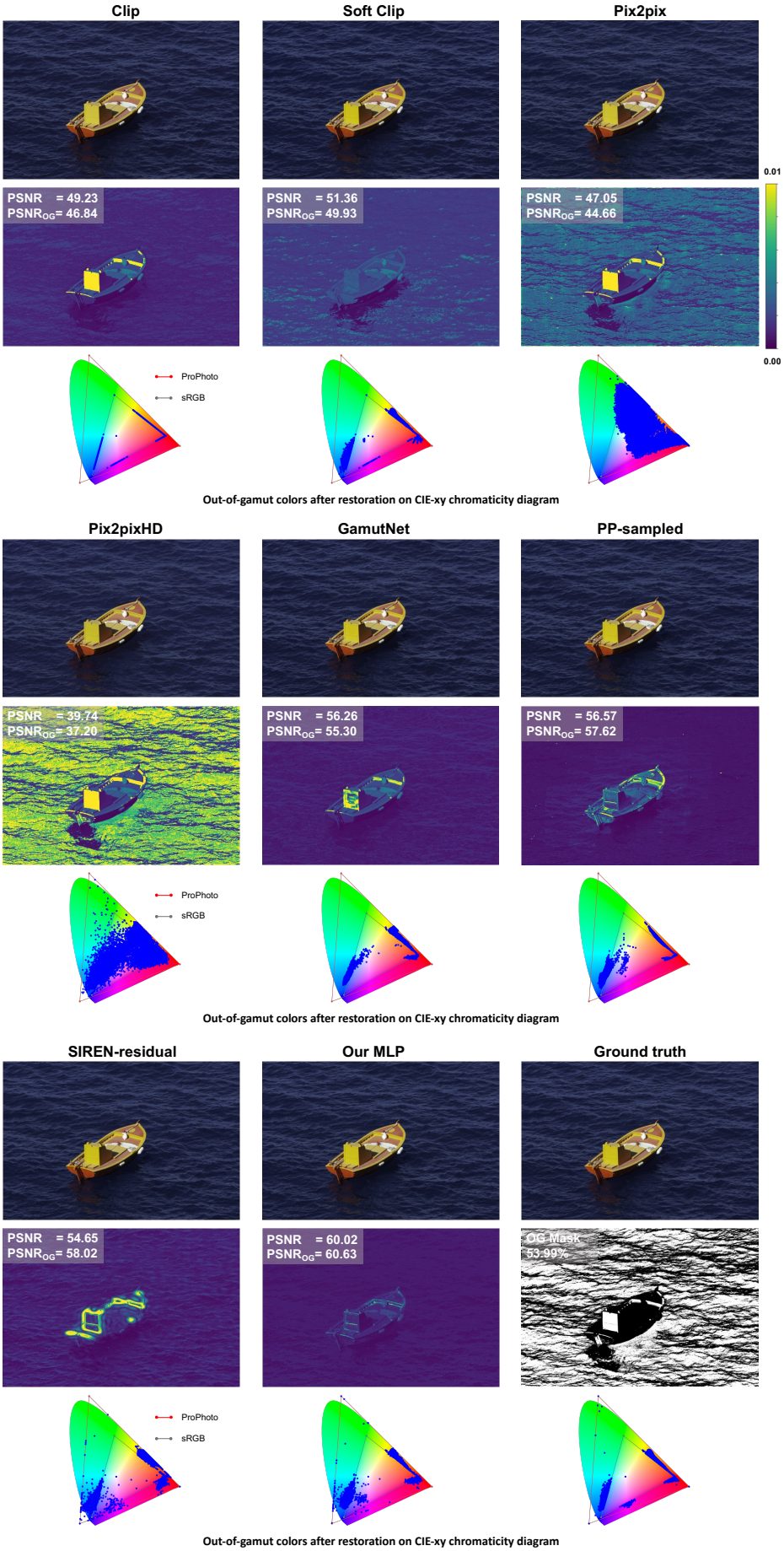
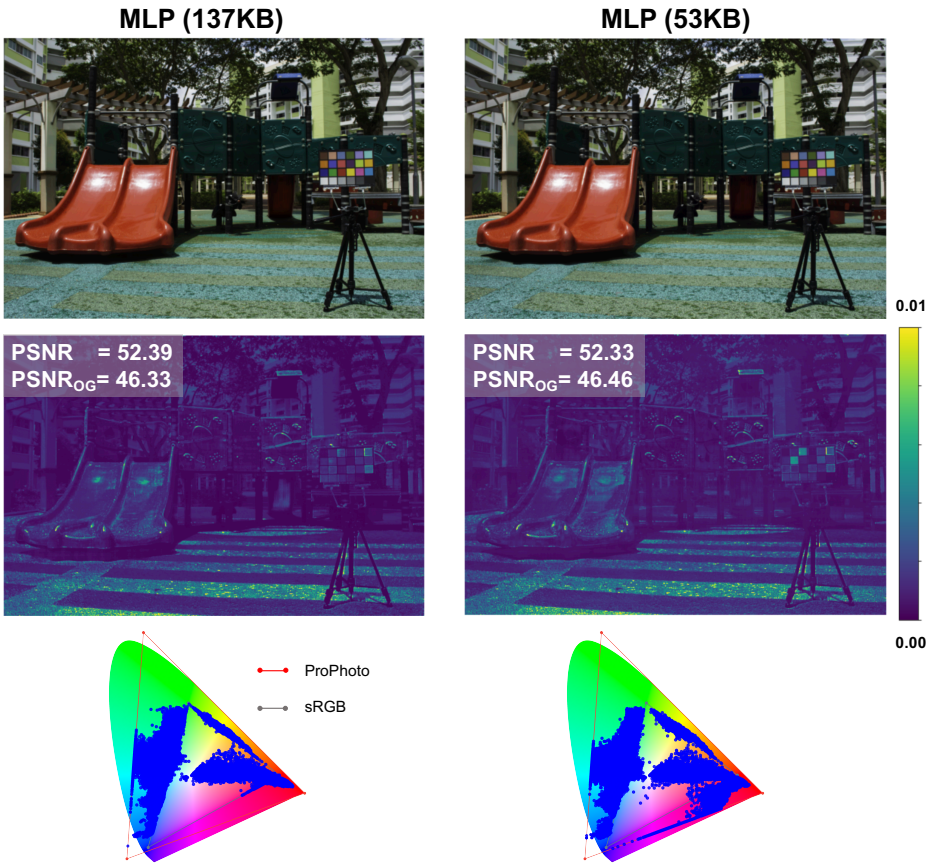


Figure 6.14: Qualitative comparisons between the predicted ProPhoto full-size output of various methods. Error maps of per-pixel RMSE and plots of out-of-gamut (OG) colors on CIE-xy chromaticity diagram with the gamuts of sRGB and ProPhoto are shown.



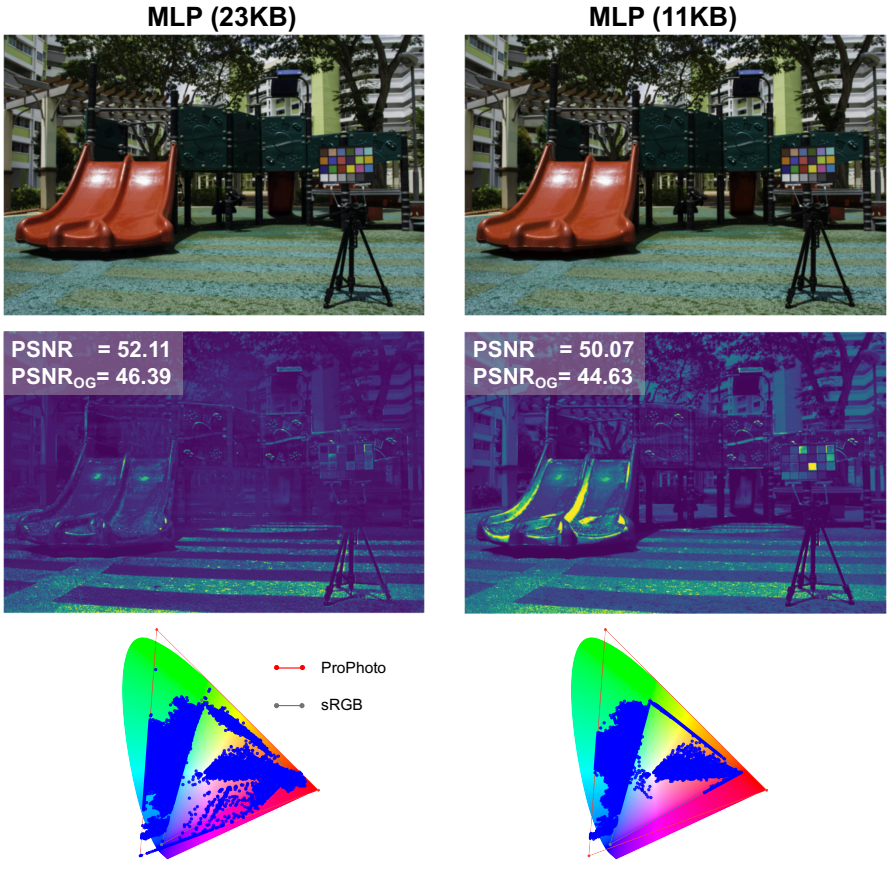


Figure 6.15: Qualitative comparisons between the predicted ProPhoto full-size output of Clip and our various versions of MLP (137 KB, 53 KB, 23 KB, and 11 KB).

Method	RMSE↓	RMSE OG↓	PSNR↑	PSNR OG↑
MLP (23 KB) [xy]	0.0037	0.0048	48.57	46.38
MLP (23 KB) [RGB]	0.0028	0.0045	51.10	46.98
MLP (23 KB) [xyRGB]	**0.0021**	**0.0031**	**53.65**	**50.04**

Table 6.2: This table shows the results of our variant MLPs with various types of inputs: *xy*, *RGB*, *xyRGB*. RMSE and PSNR for the whole image as well as OG pixels are computed and averaged over 200 test images.

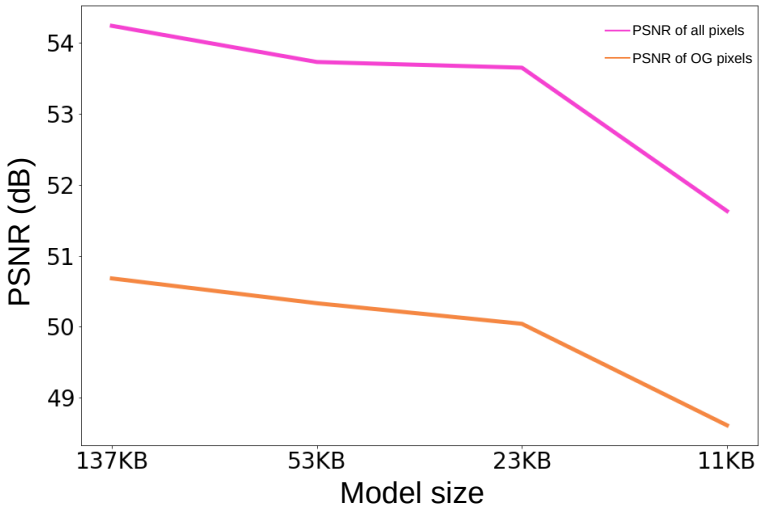


Figure 6.16: Ablation examining MLP model sizes. The 23 KB MLP provided good results for model size.

Method	**RMSE↓**	**RMSE OG↓**	**PSNR↑**	**PSNR OG↑**
MLP (23KB) w/o enc.	0.1467	0.1547	16.67	16.21
MLP (23KB) w/ enc.	**0.0021**	**0.0031**	**53.65**	**50.04**

Table 6.3: This table shows the results of our variant MLPs with and without using the encoding function. RMSE and PSNR for the whole image as well as OG pixels are computed and averaged over 200 test images.

7 Conclusions and Future Work

Color space conversion involves translating color values between different color spaces, each defined by its own color gamut (the range of visible colors it can represent). Commonly, images are initially captured in the sRGB color space, which is designed for older display technologies and covers only about 35% of visible colors. Newer color spaces like Adobe RGB and ProPhoto RGB offer much wider gamuts, suitable for modern displays and editing software. Many devices and applications still default to the limited sRGB, which poses challenges in restoring wide-gamut color values from a small-gamut image due to the loss of color information in a gamut compression stage.

We propose three methods to address this issue based on our understanding of camera pipeline. Firstly, we propose to build a deep neural network GamutNet, based on the notable deep architecture called U-Net, that learn a mapping from small-gamut colors to wide-gamut color values. Our findings indicate that GamutNet can effectively recover many of the original ProPhoto RGB values. However, it faces challenges with generalization issues, such as images that are rendered in different styles in the training set or images that are overly saturated. We then propose two additional strategies that use metadata, which represents information about the original wide-gamut color values, embedded in small-gamut outputs to aid in recovering the wide-gamut values later on. The second method we propose is to embed a small amount of original color information at capturing time,

which can help to build a machine learning model, based on polynomial functions, when we want to recover the loss of color. The function we implement can be applied globally over an image or locally depending on specific context of an image. Finally, we propose a strategy that embeds a lightweight neural network (MLP) into sRGB images instead of a sample of original wide-gamut values. This MLP is trained on pairs between ground-truth and clipped color values during the compression phase and uses that neural network to predict the ground-truth values in the expansion phase. In other words, this MLP is a compact representation of the color values in the wide-gamut images. This method stands out for its innovation, as it uses a consistent footprint for images of any size, even those with very large dimensions. It manages to store the MLP model's parameters within a JPEG image's comment field, requiring only about 20 Kilobytes of space. This means that regardless of the image's size, even for dimensions as large as 4000×6000, the storage footprint remains approximately 20 KB. Additionally, we have shown that this MLP can be trained exceptionally quickly, in under 2 seconds, to deliver top-notch results in gamut restoration. This efficiency is achieved through the use of meta-learning for improved parameter initialization and a specialized MLP implementation from NVIDIA.

While this study demonstrates the efficacy of three different approaches to recover the loss of original colors, there are several ways to improve these algorithms for future research to explore. It would be beneficial to carry out a user study comparing the recovered colors and original ones, which this study could not perform due to the lack of wide-gamut screens. Additionally, developing more efficient encoding functions or resampling methods could significantly improve the performance of our GamutMLP. Finally, our approaches could be applied to other image processing challenges with similar characteristics. For instance, they could be utilized to recover High Dynamic Range (HDR) images from Standard Dynamic Range (SDR) images. Other potential applications include restoring

colors in images compressed with lossy formats like JPEG, where color information may be lost due to compression artifacts, recovering original colors in video frames that have undergone color grading, reconstructing the original colors in scanned film negatives where the scanning process has altered the colors, correcting color distortions introduced during the conversion of images from one color space to another, and restoring original colors in satellite images that have been affected by atmospheric interference. These examples highlight the broader applicability and versatility of our methods in various image and color restoration tasks where the original colors are known but lost during processing.

References

[1] *UGRA GAMCOM Version 1.1 – Program for the Color Gamut Compression and for the comparison of calculated and measured values*, 1995.

[2] *Adobe RGB Color Image Encoding*, 1998 (Last accessed April, 2022).

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[4] Mahmoud Afifi, Abdelrahman Abdelhamed, Abdullah Abuolaim, Abhijith Punnappurath, and Michael S Brown. Cie xyz net: Unprocessing images for low-level computer vision tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[5] Mahmoud Afifi, Abhijith Punnappurath, Abdelrahman Abdelhamed, Hakki Can Karaimer, Abdullah Abuolaim, and Michael S Brown. Color temperature tuning: Allowing accurate post-capture white-balance editing. In *Color Imaging Conference (CIC)*, 2019.

[6] Youssef Alami Mejjati, Christian Richardt, James Tompkin, Darren Cosker, and Kwang In Kim. Unsupervised attention-guided image-to-image translation. *Advances in Neural Information Processing Systems*, 31, 2018.

[7] Hyrum Anderson, Eric Garcia, and Maya Gupta. Gamut expansion for video and image sets. In *14th International Conference of Image Analysis and Processing-Workshops (ICIAPW)*. IEEE, 2007.

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017.

[9] Raja Bala, Ricardo Dequeiroz, Reiner Eschbach, and Wencheng Wu. Gamut mapping to preserve spatial luminance variations. *Journal of Imaging Science and Technology*, 2001.

[10] Raja Balasubramanian, Ricardo deQueiroz, Reiner Eschbach, and Wencheng Wu. Gamut mapping to preserve spatial luminance variations. In *Color and Imaging Conference (CIC)*, 2000.

[11] Nikola Banic and Sven Loncaric. Unsupervised learning for color constancy. *CoRR*, 2017.

[12] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[13] Jacob Beck. *Surface color perception*. Cornell University Press, 1972.

[14] Gustav J Braun. A paradigm for color gamut mapping of pictorial images. *Ph.D. Thesis, Rochester Institute of Technology*, 1999.

[15] Gustav J. Braun and Mark D. Fairchild. Image lightness rescaling using sigmoidal contrast enhancement functions. In Giordano B. Beretta and Reiner Eschbach, editors, *Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts IV*, volume 3648. International Society for Optics and Photonics, SPIE.

[16] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2011.

[17] Stacey E Casella, Rodney L Heckaman, Mark D Fairchild, and Masato Sakurai. Mapping standard image content to wide-gamut displays. In *Color and Imaging Conference (CIC)*, 2008.

[18] Ayan Chakrabarti, Daniel Scharstein, and Todd E. Zickler. An empirical camera model for internet color vision. In *British Machine Vision Conference*, 2009.

[19] Ayan Chakrabarti, Ying Xiong, Baochen Sun, Trevor Darrell, Daniel Scharstein, Todd Zickler, and Kate Saenko. Modeling radiometric uncertainty for vision with tone-mapped color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2185–2198, 2014.

[20] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021.

[21] Dongliang Cheng, Dilip K Prasad, and Michael S Brown. Illuminant estimation for color constancy: Why spatial-domain methods work and the role of the color distribution. *Journal of the Optical Society of America A*, 31(5):1049–1058, 2014.

[22] International Electrotechnical Commission. IEC 61966-2-1 default RGB colour space – sRGB. *IEC*, 1999.

[23] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato. RAISE: A raw image dataset for digital image forensics. In *ACM Multimedia*, 2015.

[24] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022.

[25] Thomas Müller et al. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, July 2022.

[26] Mark D. Fairchild. *Color Appearance Models*. John Wiley Sons, Ltd, 2013.

[27] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[28] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355, 2021.

[29] Han Gong, Graham D Finlayson, Maryam M Darrodi, and Robert B Fisher. Rank-based radiometric calibration. In *Color and Imaging Conference*, 2018.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[31] Ryan D. Gow, David Renshaw, Keith Findlater, Lindsay Grant, Stuart J. McLeod, John Hart, and Robert L. Nicol. A comprehensive tool for modeling cmos image-sensor-noise performance. *IEEE Transactions on Electron Devices*, 2007.

[32] Rodney L Heckaman and James Sullivan. 18.3: Rendering digital cinema and broadcast tv content to wide gamut display media. In *SID Symposium Digest of Technical Papers*. Wiley Online Library, 2011.

[33] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021.

[34] Patrick G Herzog and Michael Müller. Gamut mapping using an analytical color gamut representation. In *Color Imaging: Device-Independent Color, Color Hard Copy, and Graphic Arts II*. SPIE, 1997.

[35] Tomotaka Hirokawa, Masao Inui, Toyoshi Morioka, and Yoshihiko Azuma. A psychophysical evaluation of a gamut expansion algorithm based on chroma mapping ii: Expansion within object color data bases. In *NIP & Digital Fabrication Conference*. Society for Imaging Science and Technology, 2007.

[36] Guowei Hong, M Ronnier Luo, and Peter A Rhodes. A study of digital camera colorimetric characterization based on polynomial modeling. *Color Research & Application*, 26(1):76–84, 2001.

[37] T Hoshino. A preferred color reproduction method for the hdtv digital still image system. In *IS&T Symposium on Electronic Photography*, 1991.

[38] Toru Hoshino. Color estimation method for expanding a color image for reproduction in a different color gamut, May 31 1994. US Patent 5,317,426.

[39] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[40] Adobe Inc. Photoshop. Last accessed April, 2022.

[41] Apple Inc. Colorsync utility. Last accessed April, 2022.

[42] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[43] Johnson A. J. Perceptual requirements of digital picture processing. *IARAIGAI symposium*, 1980.

[44] Lammens J, Ján Morovic, Nielsen M, and Huanzhao Zeng. Adaptive re-rendering of cmyk separations for digital printing. In *AIC*, 2005.

[45] Sara J. J. The automated reproduction of pictures with nonreproducible colors. *Ph.D. Thesis, Massachusetts Institute of Technology*, 1984.

[46] AJ Johnson et al. Colour appearance research for interactive system management and applications—carisma. *Work Package*, 1992.

[47] Byoung-Ho Kang, Jan Morovic, M Ronnier Luo, and Maeng-Sub Cho. Gamut compression and extension algorithms based on observer experimental data. *ETRI Journal*, 2003.

[48] Hakki Can Karaimer and Michael S. Brown. A software platform for manipulating the camera imaging pipeline. In *Proceedings of the European Conference on Computer Vision*, 2016.

[49] N Katoh and M Ito. Gamut compression for computer generated images (ii). In *Color and Imaging Conference (CIC)*, 1996.

[50] Moon-Cheol Kim, Yoon-Cheol Shin, Young-Ran Song, Sang-Jin Lee, and Il-Do Kim. Wide gamut multi-primary display for hdtv. In *Conference on Colour in Graphics, Imaging, and Vision*. Society for Imaging Science and Technology, 2004.

[51] Seon Joo Kim, Hai Ting Lin, Zheng Lu, Sabine Süsstrunk, Stephen Lin, and Michael S. Brown. A new in-camera imaging model for color computer vision and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2289–2302, 2012.

[52] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[53] Fayez Lahoud, Bin Jin, Maria Segovia, and Sabine Süsstrunk. Keyword-based image color re-rendering with semantic segmentation. In *IEEE International Conference on Image Processing (ICIP)*, 2017.

[54] Yeong-Kang Lai and Shu-Ming Lee. Wide color-gamut improvement with skin protection using content-based analysis for display systems. *Journal of Display Technology*, 9(3):146–153, 2012.

[55] Hoang Le, Mahmoud Afifi, and Michael S Brown. Improving color space conversion for camera-captured images via wide-gamut metadata. *Color and Imaging Conference*, 2020.

[56] Hoang Le, Taehong Jeong, Abdelrahman Abdelhamed, Hyun Joon Shin, and Michael S Brown. GamutNet: Restoring wide-gamut colors for camera-captured images. *Color and Imaging Conference*, 2021.

[57] Jaeho Lee, Jihoon Tack, Namhoon Lee, and Jinwoo Shin. Meta-learning sparse implicit neural representations. *Advances in Neural Information Processing Systems*, 34:11769–11780, 2021.

[58] Yihui Li, Gang Song, and Hua Li. A multilevel gamut extension method for wide gamut displays. In *2011 International Conference on Electric Information and Control Engineering*. IEEE, 2011.

[59] Ce Liu, Richard Szeliski, Sing Bing Kang, C. Lawrence Zitnick, and William T. Freeman. Automatic estimation and removal of noise from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[60] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663, 2020.

[61] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10551–10560, 2019.

[62] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. *Advances in Neural Information Processing Systems*, 31, 2018.

[63] Yu-Lun Liu, Wei-Sheng Lai, Yu Sheng Chen, Yi-Lung Kao, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. Single-image HDR reconstruction by learning to reverse the camera pipeline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[64] Lindsay MacDonald and Jan Morovic. Assessing the effects of gamut compression in the reproduction of fine art paintings. In *Color and Imaging Conference (CIC)*. Society for Imaging Science and Technology, 1995.

[65] Lindsay MacDonald, Ján Morovič, and David Saunders. Evaluation of colour fidelity for reproductions of fine art paintings. *Museum Management and Curatorship*, 14(3):253–281, 1995.

[66] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021.

[67] John Meyer and Brian Barth. Color gamut matching for hard copy. In *Applied Vision Topical Meeting*, 1989.

[68] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for

view synthesis. In *Proceedings of the European Conference on Computer Vision*, 2020.

[69] Nathan Moroney, Mark D Fairchild, Robert W.G. Hunt, Changjun Li, M Ronnier Luo, and Todd Newman. The CIECAM02 color appearance model. In *Color and Imaging Conference*, 2002.

[70] Ján Morovic. To develop a universal gamut mapping algorithm. In *Ph.D. Thesis, condensed format edition, University of Derby*, 1998.

[71] Jan Morovic. Guidelines for the evaluation of gamut mapping algorithms. *Publications-Commission Internationale de l´Eclairage CIE*, 2003.

[72] Ján Morovič. *Color gamut mapping.* John Wiley & Sons, 2008.

[73] Jan Morovic. *CIE Guidelines for Evaluation of Gamut Mapping Algorithms: Summary and Related Work (Pub. 156)*, pages 1–6. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[74] Ján Morovic and M Ronnier Luo. Developing algorithms for universal colour gamut mapping. *Colour Imaging: Vision and Technology*, 1999.

[75] Jan Morovic and Yu Wang. A multi–resolution, full–colour spatial gamut mapping algorithm. In *Color and Imaging Conference*, 2003.

[76] Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. Real-time neural radiance caching for path tracing. *ACM Transactions on Graphics*, 40(4):36:1–36:16, August 2021.

[77] Shigeki Nakauchi, Satoshi Hatanaka, and Shiro Usui. Color gamut mapping based on a perceptual image difference measure. *Color Research & Application*, 1999.

[78] Shigeki Nakauchi, Masahiro Imamura, and Shiro Usui. Color gamut mapping by optimizing perceptual image quality. In *Color and Imaging Conference*, 1996.

[79] Seonghyeon Nam and Seon Joo Kim. Modelling the scene dependent imaging in cameras with a deep neural network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017.

[80] Seonghyeon Nam, Abhijith Punnappurath, Marcus A. Brubaker, and Michael S. Brown. Learning srgb-to-raw-rgb de-rendering with content-aware metadata. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[81] László Neumann and Attila Neumann. Gamut clipping and mapping based on the coloroid system. In *Conference on Colour in Graphics, Imaging, and Vision*. Society for Imaging Science and Technology, 2004.

[82] Rang M. H. Nguyen and Michael S. Brown. RAW image reconstruction using a self-contained sRGB-JPEG image with only 64 KB overhead. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.

[83] Rang MH Nguyen and Michael S Brown. Raw image reconstruction using a self-contained sRGB–JPEG image with small memory overhead. *International Journal of Computer Vision*, 126(6):637–650, 2018.

[84] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, 2018.

[85] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5379–5389, 2019.

[86] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019.

[87] Society of Motion Picture and Television Engineers. SMPTE Recommended Practice - D-Cinema Quality — Reference Projector and Environment. *RP 431-2:2011*, pages 1–14, 2011.

[88] Laihanen P. Colour reproduction theory based on the principles of colour science. *IARAIGAI Conference Proceedings – Advances in Printing Science and Technology*, 1987.

[89] Hao Pan and Scott Daly. P-49: a gamut-mapping algorithm with separate skin and non-skin color preference controls for wide-color-gamut tv. In *SID Symposium Digest of Technical Papers*. Wiley Online Library, 2008.

[90] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7668–7677, 2019.

[91] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[92] Abhijith Punnappurath and Michael S. Brown. Spatially aware metadata for raw reconstruction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[93] Buckley R. R. Reproducing pictures with non–reproducible colors. MSc. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1978.

[94] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[95] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, 2019.

[96] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, Cham, 2015. Springer International Publishing.

[97] Viggiano J. A. S. and Wang J. A comparison of algorithms for mapping color between media of differing luminance ranges. *Technical Association of The Graphic Arts Proceedings*, 1992.

[98] K Schläpfer. Color gamut compression—correlations between calculated and measured values. *IFRA Project, EMPA, Switzerland*, 1994.

[99] Tamar Rott Shaham, Michaël Gharbi, Richard Zhang, Eli Shechtman, and Tomer Michaeli. Spatially-adaptive pixelwise networks for fast image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[100] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, 2020.

[101] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 431–440, 2020.

[102] Kevin E. Spaulding, Geoffrey J. Woolfe, and Edward J. Giorgianni. Reference input/output medium metric rgb color encodings. In *Color and Imaging Conference*, 2000.

[103] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022.

[104] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[105] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, 2020.

[106] William E Wallace and Maureen C Stone. Gamut mapping computer-generated imagery. In *Image Handling and Reproduction Systems Integration*. SPIE, 1991.

[107] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[108] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *Advances in Neural Information Processing Systems*, 33:17721–17732, 2020.

[109] Zhenyi Wang and Olga Veksler. Location augmentation for cnn. *arXiv preprint arXiv:1807.07044*, 2018.

[110] Liwen Wu, Jae Yong Lee, Anand Bhattad, Yu-Xiong Wang, and David Forsyth. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16200–16209, 2022.

[111] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021.

[112] Lu Yuan and Jian Sun. High quality image reconstruction from raw and jpeg image pair. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2158–2165, 2011.

[113] Syed Waqas Zamir, Javier Vazquez-Corral, and Marcelo Bertalmío. Perceptually-based gamut extension algorithm for emerging wide color gamut display and projection technologies. In *SMPTE 2016 Annual Technical Conference and Exhibition*. SMPTE, 2016.

[114] Syed Waqas Zamir, Javier Vazquez-Corral, and Marcelo Bertalmío. Gamut extension for cinema. *IEEE Transactions on Image Processing*, 2017.

[115] Syed Waqas Zamir, Javier Vazquez-Corral, and Marcelo Bertalmio. Vision models for wide color gamut imaging in cinema. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Nov 2019.

[116] Syed Waqas Zamir, Javierzamir2019vision Vazquez-Corral, and Marcelo Bertalmío. Gamut extension for cinema. *IEEE Transactions on Image Processing*, 26(4):1595–1606, 2017.

[117] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2223–2232, 2017.