10.2352/EI.2022.34.15.COLOR-284

Developing a gamut mapping method for a novel inkjet printer*

Baekdu Choi[1], Sige Hu[1], Rain Guo[2], White He[2], Davi He[2], George Chiu[1] and Jan P. Allebach[1]
[1] **Electronic Imaging Systems Laboratory, School of Electronic and Computer Engineering, Purdue University, West Lafayette, IN 47907, US; choi504, hu539, gchiu, allebach@purdue.edu**
[2] **Sunvalleytek International Inc., Shenzhen, CHINA**

Abstract

Gamut mapping algorithms (GMAs) map all the colors within the input image to colors reproducible with a printer. In this paper we discuss a gamut mapping algorithm we've developed for a novel inkjet nail printer. The algorithm we used previously for this suffers from visible desaturation since it only exploits the part of the printer gamut that overlaps with the sRGB gamut. To solve this issue, we add a step we call gamut alignment, which enables the printer to fully exploit the entire printer gamut. We show digitally simulated gamut mapped images to illustrate that the proposed GMA indeed produced better saturated gamut mapped images.

Introduction

Printing using any halftoning algorithm with a three colorant printer assumes that the input images have been converted to *CMY* (cyan, magenta, yellow) color space beforehand. In practice, this is not straightforward due to two reasons. First, the colorants used in printers do not have the same colors and thus the color space transformation varies with each device. Second, even if we know the color of the colorants in some device-independent color space, there is no guarantee that the color we wish to reproduce can be reproduced using the device, since the input color could be outside the printer's gamut.

Color management addresses these issues using two steps. First, the colors within the input image are mapped to colors inside the printer gamut. This process is known as *gamut mapping*. Second, after the colors are mapped to be inside the printer gamut, they are converted to the *CMY* color space for the specific printer, using the *inverse table method* module. In this paper, we focus our discussion mainly to the gamut mapping module.

This paper is organized as follows. First we introduce the gamut mapping algorithm (GMA) from [1] which our algorithm was based on and discuss the motivation for our algorithm. We then continue the discussion to introduce the main contribution in this paper, gamut alignment, and its implementation details. Lastly, we present some simulated gamut mapping results to illustrate the advantages of our algorithm.

YyCxCz color space

Gamut mapping is typically performed in a *perceptually uniform* color space. Perceptual uniformity refers to the property of the color space that a unit distance in the color space for any direction should represent the same perceived difference for human observers ([2], p.113). A commonly used perceptually uniform color space for gamut mapping is the CIE $L^*a^*b^*$ color space, but it is known to cause incorrect tone reproduction with halftone images [3]. Because of this, we choose the $YyCxCz$ color space [3] for our work, which is a linearized version of the CIE $L^*a^*b^*$ color space.

Analogous to CIE $L^*a^*b^*$, the $YyCxCz$ color space is also an opponent color space. Specifically, the Y_y component is proportional to the lightness, whereas the C_x and C_z channels form red-green and yellow-blue opponent color channels, respectively [3]. Hue h and chroma ch in $YyCxCz$ color space can also be defined as below.

$$h = \tan^{-1}\frac{C_z}{C_x},\ ch = \sqrt{C_x^2 + C_z^2} \tag{1}$$

Baseline gamut mapping algorithm

Our gamut mapping algorithm (GMA) was developed based on that in [1]. The GMA in [1] performs gamut mapping in multiple steps. First, the lightness component of the source gamut is compressed so that its range matches that of the destination gamut (Figure 1). We modified this step of the GMA in [1] slightly for our baseline. First, we only perform lightness compression when the lightness range of the source gamut is not contained in that of the destination gamut, to avoid stretching the lightness instead of compressing. Furthermore, the black point of the sRGB gamut $(Y_y, C_x, C_z) = (-16, 0, 0)$ is added to the source gamut before checking the lightness range if the source gamut only contains light colors, to avoid the colors being mapped to excessively dark colors. After that, the source and the destination gamut are rotated and aligned with respect to the Y_y axis, and both gamuts are divided into hue slices, defined as all the points with hue values within some range $[h_k, h_k + \Delta h)$ where $k = 0, \cdots, 71, \Delta h = 5°$. For the destination gamut, the white point is defined as the color of the media without any colorants, whereas the black point is defined as the color of the print when the maximum amount of colorants allowed by the printing system has been printed.

For compression, we use a piecewise linear compression, also known as soft compression, following [4] given as

$$c = \begin{cases} (1-\lambda)\frac{c_o^M}{c_i^M}c_i + \lambda c_i & 0 \le c_i < c_o^M \\ (1-\lambda)\frac{c_o^M}{c_i^M}c_i + \lambda c_o^M & c_o^M \le c_i \le c_i^M \end{cases} \tag{2}$$

where c_i is the input value of the color component to be compressed, c_o^M and c_i^M are the maximum value of the color component for the destination and source gamut, respectively. λ is set to be 0.75.

* Research supported by Sunvalleytek Internation Inc., Shenzhen, CHINA

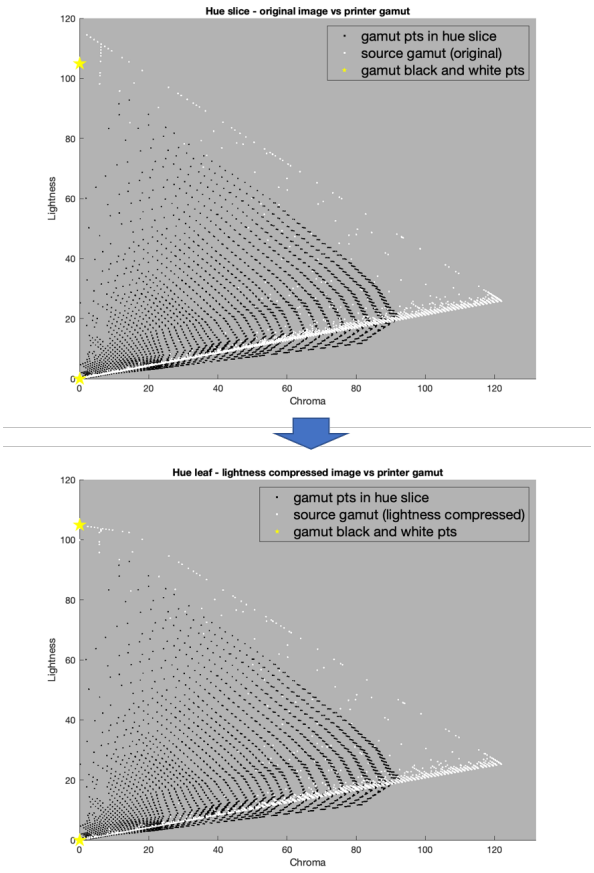

Figure 1. *Illustration of lightness compression step.*

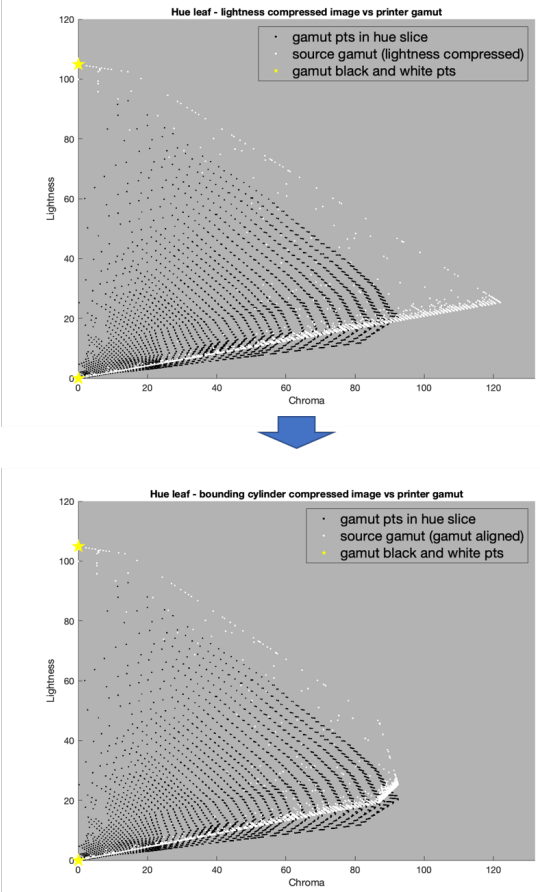

Figure 2. *Illustration of bounding cylinder compression step.*

Next, the source gamut is compressed towards the Y_y axis so that it fits inside the bounding cylinder of the destination gamut. After dividing the source and destination gamut into hue slices, we approximate the points in the same hue slice as being inside the same 2-D plane which contains the Y_y axis and the chroma axis. This allows us to refine the bounding cylinder compression as compressing the chroma values to be under the maximum chroma value of the destination gamut in the hue slice (Figure 2).

After that, each point in the source gamut is compressed towards the hue slice center. The hue slice center is defined as $(ch, Y_y) = (0, \frac{Y_{max}+Y_{min}}{2})$ for all hue slices, where Y_{max} and Y_{min} are the maximum and minimum lightness values of the destination gamut after rotation and shift, respectively. For the hue slice center compression, first we find the convex hull of the destination gamut for each hue slice, using the gift wrapping algorithm [5]. Since the source gamut points tend to cluster together rather than being evenly distributed, it requires a large amount of computation to find the exact maximum distance from the hue slice center to the source gamut points along a fixed direction. Instead, we approximate the maximum distance by dividing the Y_y axis - chroma axis plane into 36 *altitude bins* according to the direction from the hue slice center, as illustrated in Figure 3. For each altitude bin, it is assumed that the maximum distance from the hue slice center to the source gamut points in the bin is constant. As a preprocessing step, we go through the source gamut points and record the maximum distance for each altitude bin in a look-up table. After that, when performing hue slice center compression (Figure 4), we can easily find the maximum distance towards the same direction by simply referencing the look-up table.

The step of compressing all the source gamut points toward the hue slice center guarantees that all points in source gamut are mapped to be inside the destination gamut. The last step is to rotate and shift the source gamut back so that the neutral axis of the gamut mapped source gamut aligns with that of the original destination gamut.

While the baseline GMA in general generates good quality gamut mapped images, it often results in the source images getting severely desaturated, as can be seen in Figure 5. To understand why this is happening, first note that the baseline GMA does not move the colors outside the *sRGB* gamut. That is, the parts of the printer gamut that do not overlap with the *sRGB* gamut do not get utilized after gamut mapping. This is not ideal since a typical hue slice looks like Figure 6.

In Figure 6, the black points illustrate the *sRGB* gamut points, whereas the blue points illustrate the printer gamut points. Note that there is a rather large region included in the printer gamut but is outside the *sRGB* gamut. Since saturation is defined as the colorfulness of the color relative to its brightness ([6], p.13), i.e., the ratio between the chroma and the lightness of the color, this region includes more saturated colors of the printer gamut in

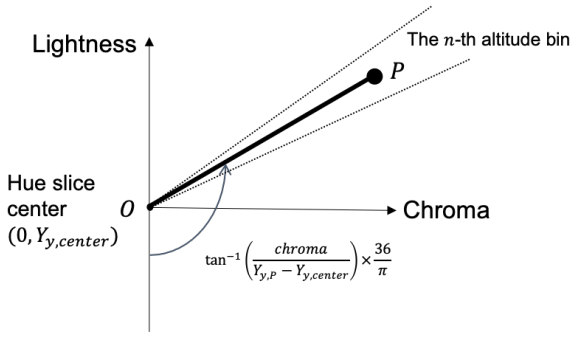

Figure 3. *Illustration of the n-th altitude bin which includes the point P with lightness value $Y_{y,P}$ and chroma value ch_P. The index of altitude bin n where the point P belongs can be found as $n = \lfloor \tan^{-1}(\frac{ch_P}{Y_{y,P} - Y_{y,center}} \times \frac{36}{\pi}) \rfloor$.*

the hue slice. This indicates that to get more saturated prints, the GMA should be moving the colors to be *outside* the *sRGB* gamut, which leads to our discussion in the next section of the paper.

Gamut alignment

As explained in the previous section, it is necessary that the GMA move the colors outside the *sRGB* gamut in order to get more saturated gamut mapped images. Therefore, we propose using a step we call *gamut alignment* instead of the bounding cylinder compression step in the baseline GMA.

Gamut alignment

We propose the gamut alignment step to move the source gamut colors outside the *sRGB* gamut with the following properties:

- The *sRGB* gamut maximum chroma point for a hue slice should be mapped to a point along the direction from the hue slice center to the printer gamut maximum chroma point; so that after hue slice center compression, the point gets mapped to the printer gamut maximum chroma point. This guarantees that after hue slice center compression, if the source image included the maximum chroma point in that hue slice, the point gets mapped to the maximum chroma point of the printer gamut in the hue slice.
- The source gamut points along the neutral axis do not move. This is to avoid mapping neutral colors in the source gamut to non-neutral colors.

The gamut alignment step is performed in hue slices, so each mapping is performed within the 2-D plane that represents the hue slice. Let us shift the plane so that the hue slice center $(0, Y_{y,center})$ is the origin of the plane, and call the plane a hue slice plane. Let us denote each point in the hue slice plane using (ch, Y_s) where ch is the chroma defined in Eq. 1 and $Y_s = Y_y - Y_{y,center}$. Note that shifting the plane does not alter the chroma components since the hue slice center has chroma value of 0. Let us denote the *sRGB* gamut point with maximum chroma value in the hue slice plane as $(ch_{max}^{sRGB}, Y_{s,max}^{sRGB})$. Let us define two unit length vectors $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ in the hue slice plane as

$$\boldsymbol{u_1} = (0,1),\ \boldsymbol{u_2} = \frac{(ch_{max}^{sRGB}, Y_{s,max}^{sRGB})}{\|(ch_{max}^{sRGB}, Y_{s,max}^{sRGB})\|_2} \tag{3}$$

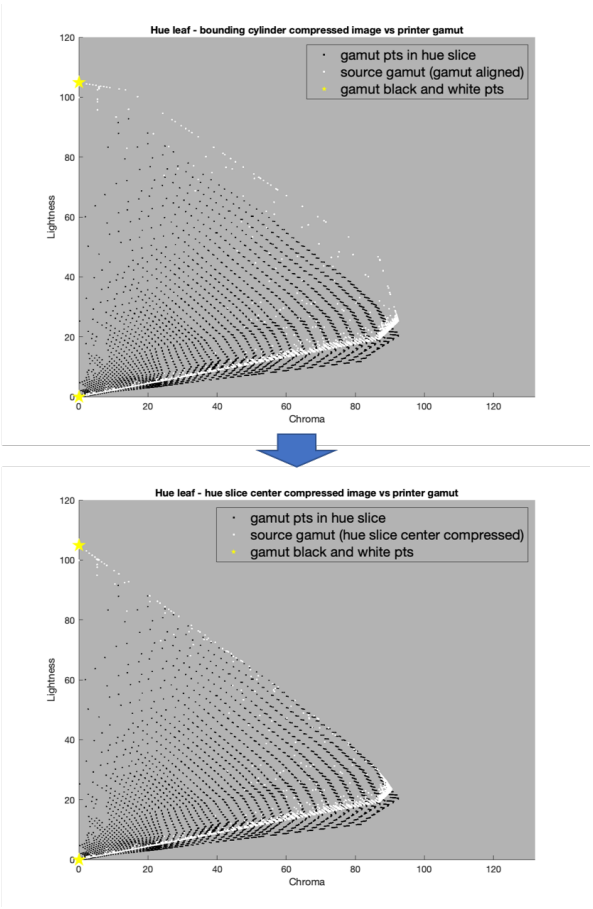

Figure 4. *Illustration of hue slice center compression step.*

Note that since $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$ are linearly independent, they span the entire hue slice plane. That is, any point $\boldsymbol{p}$ in the hue slice plane can be expressed as $\boldsymbol{p} = a_1\boldsymbol{u_1} + a_2\boldsymbol{u_2}$, where

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{u_1} & \boldsymbol{u_2} \end{bmatrix}^{-1} \boldsymbol{p} \tag{4}$$

Now, let us denote the printer gamut point with maximum chroma value in the hue slice plane as $(ch_{max}^{dest}, Y_{s,max}^{dest})$. Let us define a third unit vector $\boldsymbol{u_3}$ as

$$\boldsymbol{u_3} = \frac{(ch_{max}^{dest}, Y_{s,max}^{dest})}{\|(ch_{max}^{dest}, Y_{s,max}^{dest})\|_2} \tag{5}$$

Then, gamut alignment will map a point $\boldsymbol{p}$ in the hue slice plane to

$$\boldsymbol{p'} = a_1\boldsymbol{u_1} + a_2\boldsymbol{u_3} = \begin{bmatrix} \boldsymbol{u_1} & \boldsymbol{u_3} \end{bmatrix} \begin{bmatrix} \boldsymbol{u_1} & \boldsymbol{u_2} \end{bmatrix}^{-1} \boldsymbol{p} \tag{6}$$

Note that this mapping indeed satisfies the desired properties for moving the source gamut points. First, if $\boldsymbol{p}$ lies on the Y_y axis, $\boldsymbol{p} = a_1\boldsymbol{u_1}$, thus $\boldsymbol{p'} = \boldsymbol{p}$. Second, for the *sRGB* gamut point with maximum chroma value in the hue slice plane, $\boldsymbol{p} = a_2\boldsymbol{u_2}$, thus after mapping we have $\boldsymbol{p'} = a_2\boldsymbol{u_3}$, so the mapped point is along the direction from the hue slice center to the printer gamut's maximum chroma point in the hue slice plane. An illustration of the

Figure 5. *Illustration of desaturation issue with the baseline GMA in [1]. On the left column are the original input images, whereas on the right column are the gamut mapped images generated with the baseline GMA. The desaturation is evident in the rose petals of the top row images and in the sky of the bottom row images.*

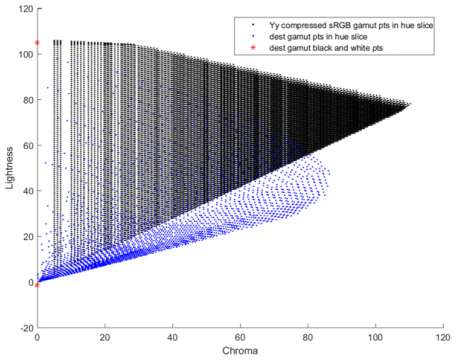

Figure 6. *A typical hue slice for sRGB and printer gamut, after lightness compression.*

gamut alignment step for a point ***p*** in the source gamut is in Figure 7.

The effect of gamut alignment step for a typical hue slice is illustrated in Figure 8. Note that in the figure, the black points illustrate the *sRGB* gamut points in the hue slice, whereas the cyan points illustrate the printer gamut points in the hue slice.

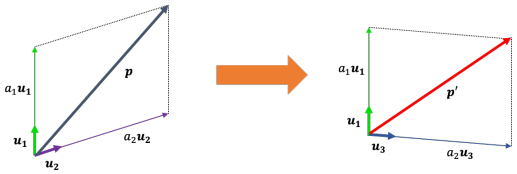

Figure 7. *Illustration of gamut alignment for a point **p** in the source gamut.*

Figure 9 illustrates the effect of the gamut alignment step on the GMA. Note that while the images in general are more saturated as expected, the bottom row images show that false contour artifacts occur when the gamut alignment step is naively implemented without any postprocessing. This leads to our discussion

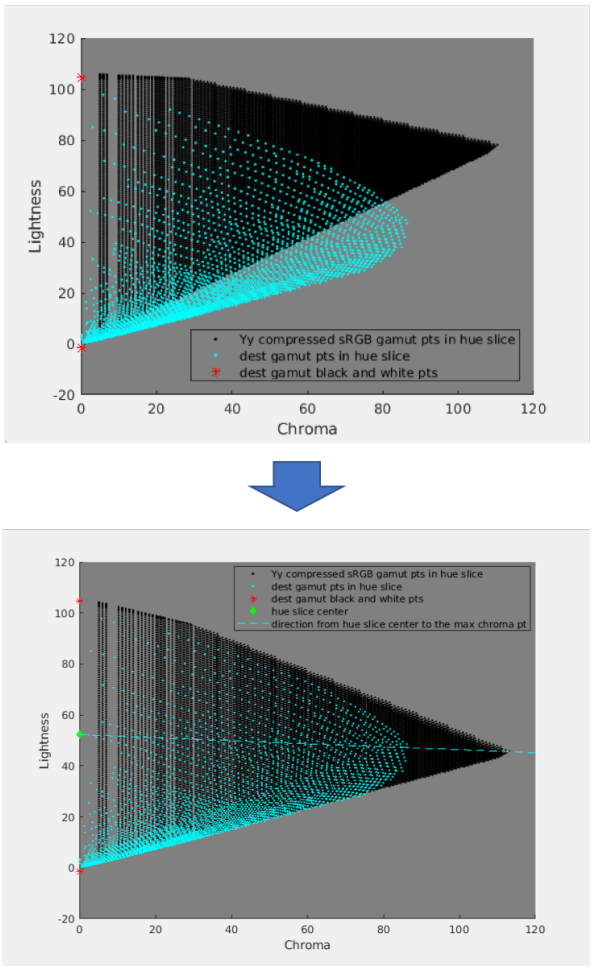

Figure 8. *A typical hue slice before and after gamut alignment.*

of blending-in in the next section.

Blending-in to address false contour artifacts

As seen in Figure 9, naively implementing gamut alignment results in false contours in the gamut mapped images. This is because of the discontinuity occurring near the edges of each hue slice. Unlike the baseline GMA, gamut alignment moves points aggressively, which makes the discontinuity significantly more visible in the resulting gamut mapped image. Furthermore, another discontinuity issue occurs due to the use of altitude bins in the hue slice center compression step, which also results in visible artifacts when used in GMA with the gamut alignment step.

To address the discontinuity issue, we propose blending-in as a postprocessing step of the hue slice center compression step of the GMA. For this, first we perform gamut mapping to the source image up to the lightness compression step. Then, we assign two hue slices per each point, instead of one, to perform the following steps as described below:

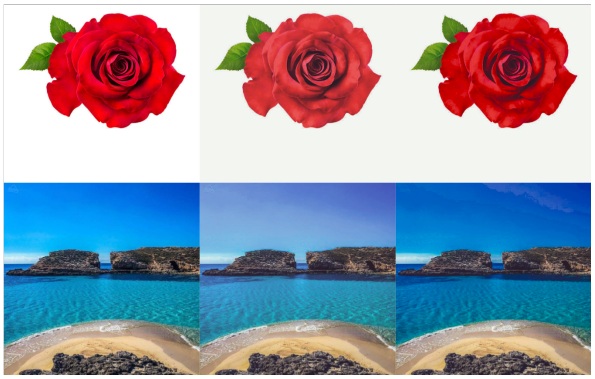

Figure 9. *Simulated gamut mapping results using naive implementation of the gamut alignment step. Left: source image, middle: baseline GMA result, right: GMA with naive gamut alignment result.*

- hue slice 1 $= \lfloor(\tan^{-1}(\frac{C_x}{C_z})+\pi)\cdot\frac{36}{\pi}\rfloor$
- $\eta = (\tan^{-1}(\frac{C_x}{C_z})+\pi)\cdot\frac{36}{\pi} - \lfloor(\tan^{-1}(\frac{C_x}{C_z})+\pi)\cdot\frac{36}{\pi}\rfloor$
- If $\eta \geq 0.5$, $\alpha = 1-\eta+0.5$, hue slice 2 = $\lfloor(\tan^{-1}(\frac{C_x}{C_z})+\pi)\cdot\frac{36}{\pi}\rfloor+1$
- If $\eta < 0.5$, $\alpha = \eta+0.5$, hue slice 2 = $\lfloor(\tan^{-1}(\frac{C_x}{C_z})+\pi)\cdot\frac{36}{\pi}\rfloor-1$

Figure 10 illustrates the step described above. In the case of the point $\boldsymbol{p}$ in Figure 10, hue slice 1 will be s_1 in the figure. Since the point is to the left with respect to the center of the hue slice s_1, $\eta < 0.5$ in this case, so hue slice 2 will be s_0 in the figure.

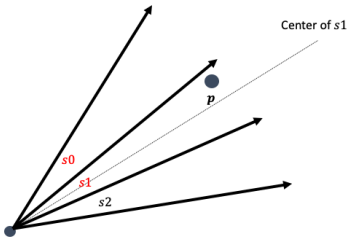

Figure 10. *Illustration of finding hue slice 2.*

After hue slice 1 and hue slice 2 are determined for each point in the source gamut, gamut alignment and hue slice center compression steps can be performed twice using hue slice 1 and 2 respectively. If a point $\boldsymbol{p}$ gets mapped into $\boldsymbol{p_1}$ using hue slice 1 and $\boldsymbol{p_2}$ using hue slice 2, we can find the final mapping result $\boldsymbol{p'}$ as $\boldsymbol{p'} = (1-\alpha)\boldsymbol{p_1} + \alpha\boldsymbol{p_2}$.

As explained previously, the hue slice center compression itself also has to be modified to address the discontinuity issue in the use of altitude bins. We can apply a similar strategy to address the discontinuity issue with the use of altitude bins as well, as below:

- bin 1 = $\lfloor\tan^{-1}(\frac{ch}{Y_y - Y_{y,center}})\cdot\frac{36}{\pi}\rfloor$
- $\alpha = \tan^{-1}(\frac{ch}{Y_y - Y_{y,center}})\cdot\frac{36}{\pi} - \lfloor\tan^{-1}(\frac{ch}{Y_y - Y_{y,center}})\cdot\frac{36}{\pi}\rfloor$
- If $\alpha \geq 0.5$, bin 2 = $\lfloor\tan^{-1}(\frac{ch}{Y_y - Y_{y,center}})\cdot\frac{36}{\pi}\rfloor+1$, $\beta = 1.0-\alpha+0.5$
- Otherwise, bin 2 = $\lfloor\tan^{-1}(\frac{ch}{Y_y - Y_{y,center}})\cdot\frac{36}{\pi}\rfloor-1$, $\beta = \alpha+0.5$
- Compute the two center compression results using bin 1 and bin 2 as altitude bins to get two mapped results $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$
- The final mapped result is computed as $\boldsymbol{p'} = (1-\beta)\boldsymbol{p_1} + \beta\boldsymbol{p_2}$

Figure 11 shows the results of gamut mapping with or without blending-in. Note that the false contour artifacts get suppressed when we add the blending-in steps, as expected.

Figure 11. *Simulated gamut mapping results illustrating the need for the blending-in steps. Left: GMA with naive gamut alignment, middle: GMA with gamut alignment & hue slice blending-in but without altitude bin blending-in, right: GMA with all the blending-ins.*

Results & Conclusions

In this section we present a comparison of the baseline GMA and our proposed GMA. For this purpose, a set of simulated gamut mapped images using the baseline and proposed GMA are shown in Figure 12. Note that for input images with white background, the background gets mapped to a gray color, which is the same as the measured color of the paper that we used in our experiments.

Comparing the simulated gamut mapping results, we can draw the following conclusions:

- Comparing the results from the baseline GMA and our proposed GMA, we can see that our proposed GMA provides better saturated results.
- While our proposed GMA increases saturation for the gamut mapped images, it does not overly enhance saturation and generate visually unpleasant images. For instance, the female figure's face image in the second row of Figure 12 shows that in the result from our proposed GMA, the female's skin color is still reasonable despite the overall enhanced saturation.

To summarize, we proposed a novel GMA that generates better saturated gamut mapped images compared to the baseline GMA in this paper. We first investigated the cause of desaturation in the baseline GMA and made two important observations: first, the baseline GMA does not fully utilize the printer gamut, and second, the part of the printer gamut that the baseline GMA does not utilize includes the most saturated colors within the printer gamut. To utilize the previously unused part of the

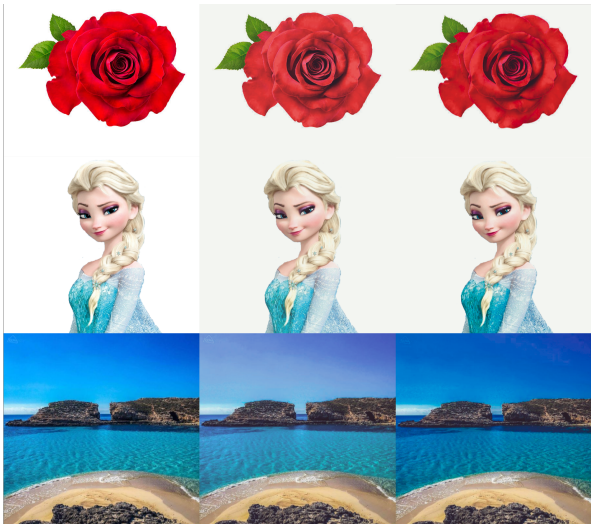

Figure 12. *Comparison of the baseline and the proposed GMA using simulated results. Left: input image, middle: baseline GMA result, right: proposed GMA result.*

printer gamut, we propose replacing the bounding cylinder compression step in the baseline GMA with a new step we call gamut alignment, which moves the points in the source gamut outside the *sRGB* gamut. This step potentially introduces false contour issues, which comes from the discontinuities due to how we divide the printer gamut into hue slices. Thus, we added multiple blending-in steps to ensure that the artifacts are successfully suppressed in the final gamut mapped images.

References

[1] J. Liu, T. Frank, Y. Ben-Shoshan, R. Ulichney, and J. P. Allebach, "NPAC FM color halftoning for the Indigo press : challenges and solutions," in *Proc. 24th Color Imag., Displaying, Process., Hardcopy, Appl. IS&T*, Burlingame, CA, USA, 2019, pp. 102–1–102–8.

[2] J. Morovič, *Color Gamut Mapping*. Chichester, West Sussex, England: John Wiley & Sons Ltd, 2008.

[3] T. J. Flohr, B. W. Kolpatzik, R. Balasubramanian, D. A. Carrara, C. A. Bouman, and J. P. Allebach, "Model-based color image quantization," in *Proc. SPIE 1913, Human Vision, Visual Processing, and Digital Display IV*, San Jose, CA, USA, 1993, pp. 270–281.

[4] R. Gentile, E. Walowit, and J. P. Allebach, "A comparison of techniques for color gamut mismatch compensation," *J. Imag. Technol.*, vol. 16, no. 5, pp. 176–181, 1990.

[5] R. A. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information Processing Letters*, vol. 2, no. 1, pp. 18–21, 1973.

[6] R. W. G. Hunt and M. R. Pointer, *Measuring Colour*. Chichester, West Sussex, England: John Wiley & Sons Ltd, 2011.

Author Biography

Baekdu Choi received his B.Sc. in electrical and computer engineering from Seoul National University, Seoul, South Korea in 2017 and is currently working on a Ph.D. in electrical and computer engineering at Purdue University, West Lafayette, IN, USA. His research mainly focuses on digital image processing, digital halftoning and color management for inkjet printers.

Sige Hu received his B.S. in Electrical and Computer Engineering from Pennsylvania State University, State College, PA, USA in 2017 and is currently pursuing Ph.D. in Electrical and Computer Engineering at Purdue University, West Lafayette, IN, USA. His research mainly focuses on image processing.

George T. Chiu is a Professor in the School of Mechanical Engineering with courtesy appointments in the School of Electrical and Computer Engineering and the Department of Psychological Sciences at Purdue University. He also serves as the Assistant Dean for Global Engineering Programs and Partnership for the College of Engineering. Dr. Chiu received the B.S. degree in Mechanical Engineering from the National Taiwan University in 1985 and the M.S. and Ph.D. degrees in Mechanical Engineering from the University of California at Berkeley, in 1990 and 1994, respectively. From September 2011 to June 2014, he served as the Program Director for the Control Systems Program at the National Science Foundation. His current research interests are mechatronics and dynamic systems and control with applications to digital printing and imaging systems, digital fabrication and functional printing, human motor control, motion and vibration perception and control. He received the 2012 NSF Director's Collaboration Award and the 2010 IEEE Transactions on Control System Technology Outstanding Paper Award. He served as the Editor-in-Chief for the IEEE/ASME Transactions on Mechatronics from 2017-19 and as the Editor for the Journal of Imaging Science and Technology from 2012-14. Dr. Chiu served on the Executive Committee of the ASME Dynamic Systems and Control Division (DSCD) from 2007 to 2014 and as the Chair of the Division from 2012-13. He is a Fellow of ASME and a Fellow of the Society for Imaging Science and Technology (IS&T).

Jan P. Allebach is Hewlett-Packard Distinguished Professor of Electrical and Computer Engineering at Purdue University. Allebach is a Fellow of the IEEE, the National Academy of Inventors, the Society for Imaging Science and Technology (IS&T), and SPIE. He was named Electronic Imaging Scientist of the Year by IS&T and SPIE, and was named Honorary Member of IS&T, the highest award that IS&T bestows. He has received the IEEE Daniel E. Noble Award, the IS&T/OSA Edwin Land Medal, and the IS&T Johann Gutenberg Prize. He is a member of the National Academy of Engineering.