An RGB Color Management Concept based on an Improved Gamut Mapping Algorithm

Ursina Caluori, Klaus Simon

Swiss Federal Laboratory for Materials Testing and Research (EMPA), Dübendorf, Switzerland

ABSTRACT

In principal, color reproduction on specific devices can be divided into transformation of color coordinates and the adaptation of colors to a device, called gamut mapping. The well established ICC color management neglects the latter aspect. This paper presents a practical implementation addressing both an efficient color transformation and a sophisticated gamut mapping.

Keywords: gamut mapping, rgb workflow, color management, device characterization, icc

1. INTRODUCTION

One of the big challenges in today's desktop publishing world is the colorimetric correct reproduction of color. Due to the limited range of colors reproducible by each device, an adaptation of the input colors, so-called gamut mapping (GM),[1] is usually involved.

15 years ago, the International Color Consortium developed an open distributed software standard (the *ICC color management system* or *ICC-CMS*) to universally transform between color spaces, in particular device specific ones. Originally, this system has been designed for prepress applications, but nowadays prepress is reduced substantially. The responsibility of the reproduction process is increasingly being shifted toward the user, namely journalists, authors, agencies, etc.

Considering the growing size and popularity of RGB color spaces, the central interest in color management changes from "color space transformation" to a "sophisticated gamut mapping".

The main drawback of ICC-CMS is that it is based on precomputed data structures (mostly lookup-tables), which means that the involved GM has to be precomputed, too. This prohibits image-dependent GM. However, recent research suggests that exactly this approach produces the visually most satisfying results.

We propose a simple, user-friendly RGB-workflow based on a sophisticated GM-algorithm. RGB input data is transformed into output device RGB data. Without further manipulation, the resulting data leads to a colorimetric correct result using a native RGB-printer or the always available "$cyan = 1 - red$" interface to a CMYK-printer (PDF Reference,[2] Chapt. 6.2). This output format explicitly addresses widespread publishing standards without ICC-CMS support, like LaTeX.

The first part of our solution is the device characterization, done by offline measurement of a set of RGB patches. This data is used to build a regular grid structure in the working color space, to support an efficient backtransform to device RGB using trilinear interpolation.

The second part is a new and innovative spatial gamut mapping algorithm. State-of-the-art algorithms (e.g. Kolas et al.,[3] Giesen et al.,[4] Kimmel et al.[5]) tend to be based on the set of colors contained in the input image (*image gamut*) instead of the whole input color space. We apply this concept to the local neighborhood of every image pixel separately, by using an edge-preserving smoothing filter to estimate the neighborhood. This allows us to avoid the explicit determination of the image gamut, which would be a nontrivial task. Furthermore, it increases the potential for visual optimization.

Further author information: (Send correspondence to U. Caluori)
U. Caluori: E-mail: ursina.caluori@empa.ch, Telephone: +41 44 823 4032

Color Imaging XIV: Displaying, Processing, Hardcopy, and Applications, edited by Reiner Eschbach
Gabriel G. Marcu, Shoji Tominaga, Alessandro Rizzi, Proc. of SPIE-IS&T Electronic Imaging
SPIE Vol. 7241, 724118 · © 2009 SPIE-IS&T · CCC code: 0277-786X/09/$18 · doi: 10.1117/12.805320

2. MOTIVATION

This paper is to understand as a reaction to ICC-CMS, which is an open, distributed software standard, based on measurements that are stored in so-called profiles. The central idea of ICC-CMS is the transform of device-specific coordinates to a device-independent color space like CIELAB or CIE-XYZ, called *profile connection space (PCS)*. This transform is stored in a profile, which usually contains nothing more than a list of source and target values (a look-up-table or LUT), which act as interpolation samples. For less complex transforms, the LUT may be replaced by a matrix, in particular in the context of monitor characterization. The standard ICC workflow comprises two central steps:

1. Transform input data to PCS.
2. Transform PCS data into output device color space.

In both steps, a profile is applied to the data by a component called *color matching method (CMM)*. This concept is *universal* in the sense that arbitrary input data can be transformed to arbitrary output data. A profile is generated by the measurement of a number of color patches. The resulting transform information is usually stored in the form of a lookup table. Applying a profile is then mostly an interpolation of the generated sampling points.

However, in the course of time, serious problems around the ICC-CMS emerged:

- Since there is no information about the kind of interpolation originally associated with the profile, the outcome of a transform is heavily dependent on the interpretation of the profile by the interpreting CMM. If, for example, one were to use the same profile on the same data on the same monitor one time running Mac OS and one time running Windows, the results would probably not look the same, because each operating system has its own CMM and will not necessarily use the same interpolation.
- The ICC profile data is static. The color transform is completely defined by the ICC profile. Because of this static structure, the induced color transform, hard-coded in the profile, it is also static. Thus, image-dependent gamut mapping is not possible.
- Several interpolation steps are involved. One for each the profile generation and application of the input profile, and the same for the output profile, totaling four interpolation steps, each introducing new potential for numerical errors and propagation thereof.
- The responsibility of choosing the right profile in the right place is left to the user.
- An open modular specification is error-prone. This is critical because in particular there is the risk of incorrect combination of the single components. Each color transform is a transaction, but in ICC-CMS there is no real transaction control, because the interpretation of a profile is left to the CMM.
- The ICC workflow is based on D50 as white point, but many RGB applications work with D65 data, i.e. there is a need for explicit conversion, which is only an approximation.

We address these problems by introducing an alternative, closed workflow and putting aside the universatility ambition. The transform to and from PCS was even recognized by the ICC as a serious problem as shown by the introduction of device link profiles, which eliminate this step. We take this approach a step further and integrate everything into one big black box, effectively giving us transaction control and the power to do sophisticated gamut mapping.

Over the last few years, the demand has shifted from prepress applications to desktop publishing and consequently RGB workflows are becoming more and more important. There is also a demand for visually better algorithms than the state-of-the-art industry standards allow for. To meet these demands, a closed RGB workflow that does not require expert knowledge on the user's side is needed.

3. THE CONCEPT

The standard workflow for the application of our concept is pictured in figure 1. When a user sends his RGB data into our program module, it first gets converted to the working color space (*WCS*, e.g. CIELAB), then the adaptation from input gamut to output gamut takes place, and finally the data is transformed to device specific RGB coordinates. In order for this application to work, several preprocessing steps are necessary.

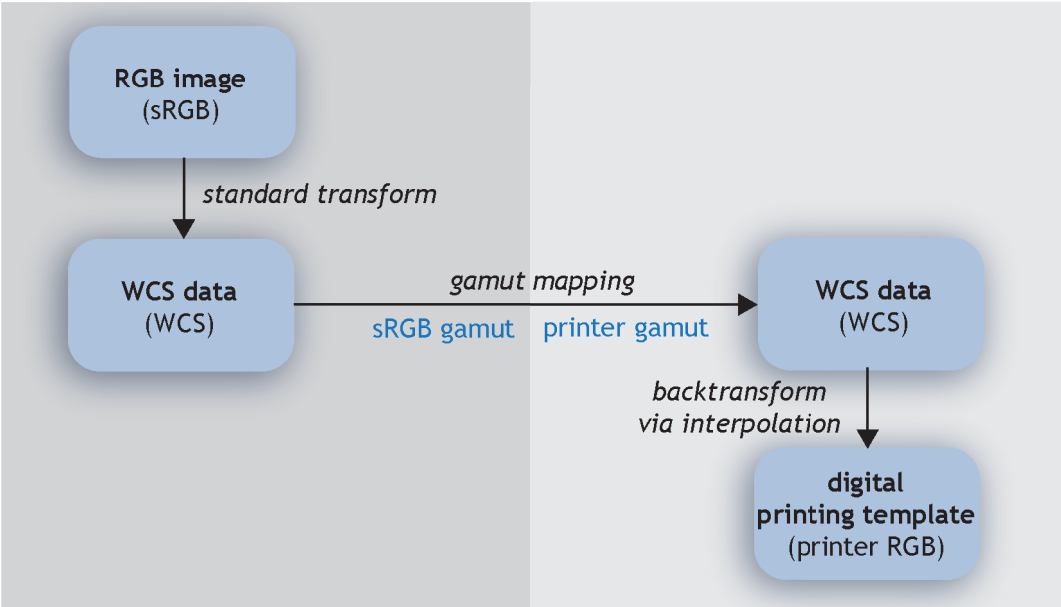

Figure 1. Essential workflow with sRGB input data.

In detail, our approach is based on the following main points, divided into a preprocessing and an application part. The preprocessing has to be done only once for a specific device:

1. An offline device characterization is done as described in section 3.1.
2. The output gamut is directly derived from the characterization of the output device. See section 3.1.1.
3. The data structure for the inverse transform (working color space coordinates to device RGB) is also derived from the characterization measurements. See section 3.1.2.

As for the application, the following steps are performed everytime an RGB data file is sent into our program module:

1. The RGB input data is converted to the working color space. The input for our color transform can be sRGB, AdobeRGB98, ECI-RGB or some other RGB compatible to CIE color spaces. The only requirement is the existence of formulas that allow an exact and efficient transformation of color data to the working color space CIELAB (or optionally IPT) without interpolation. In order to avoid unnecessary computations for D50 white point adaptation, the entire workflow is based on D65.
2. The data is mapped to the output device's gamut using a filter-based, spatial gamut mapping algorithm, that is realized as a 3D-polyhedron algorithm. This is described in detail in section 4.
3. The mapped working color space data is transformed to device RGB coordinates. This inverse transform is implemented as lattice-based interpolation. After the data has been transformed to device RGB coordinates, the resulting RGB-file can be sent to the printer without further manipulation. See section 3.1.2.

3.1 Device Characterization

The device characterization comprises two central structures. One for the estimation and storage of the gamut surface and the other for transforming the data back from the working color space to device specific coordinates. In order to build these structures, device specific measurements are required. In a first step, the RGB cube is partitioned into a regular RGB grid, with every node representing a color patch. Then these patches are printed (using "$cyan = 1 - red$" conversion) on the device being characterized and subsequently measured with the eye-one photospectrometer. Finally, the obtained source vs. target value pairs are stored in a data file.

3.1.1 Gamut description

In order to find a suitable gamut description, a segment maxima method analogous to Morovič,[6] but with higher segment resolution, is applied in the working color space. First the space, which we treat as a sphere, is partitioned into 808 segments along the equator and then each of these is further split into 404 segments from pole to pole. This structure is then used to build a new structure by combining the segments to blocks, such that every block corresponds to approximately $1\Delta E$ relative to the gamut. In the next step, the blocks get populated with the measured gamut points (one linear list of points per block). Since there are much more blocks than sampling points, we triangulate the gamut surface, such that every block contains at least one known gamut surface point. For every block, the point with the biggest radius is called the *block representative*. These representatives define the gamut surface, and are used for in-gamut queries as well as compression in gamut mapping (see section 4). All sampling points, including those not on the surface are used for building the data structure for the backtransform, described in the next section.

To find the containing block of a given point $\mathbf{p}$ we first compute segment indices i and j from the point's polar angles and then use these as indices into the array of corresponding block numbers, which is an operation in $O(1)$.

3.1.2 Backtransform

The output is organized in two steps. The first step is the offline computation of a suitable data structure for enabling fast backtransform. This is done by fitting the measured data to a regular grid in the working color space. The challenge is to find the smallest containing polyhedron, which is also known as the point-location problem. This problem can be reduced to a sorting problem. By clever sorting of the polyhedrons, a point-location query can then be performed in $O(1)$. We use this fact to build a corresponding heuristic. In fact, the sorting was already done by building the block structure described above. The fitting of the measured data to a working color space grid vertex $\mathbf{v}$ is done as follows:

1. Compute the containing block $\mathbf{b}$ of $\mathbf{v}$, as described in the previous section.
2. Find the point $\mathbf{p}_m$ in $\mathbf{b}$ with minimal radius bigger than or equal to the radius of $\mathbf{v}$.
3. Check if one of the adjacent RGB grid cubes of $\mathbf{p}_m$ contains $\mathbf{v}$ by partitioning each of the cubes into tetrahedrons and doing an inside test.
4. If found, use the resulting barycentric coordinates for the interpolation of $\mathbf{v}$, otherwise do a breadth-first search in the given RGB grid for the containing polyhedron of $\mathbf{v}$.

The grid inside the gamut is now completely defined. But in order to guarantee the reachability of every point inside the gamut, in particular those on the surface, the grid vertices just outside the gamut also have to be determined. For that purpose, every grid vertex that lies outside the gamut and has at least one in-gamut neighbor is extrapolated from the already calculated in-gamut vertices.

The second step is the actual backtransform, which happens at the very end of the workflow. This is done by finding the grid cube that contains the point $\mathbf{p}_t$ to be transformed and then performing a trilinear interpolation. The containing cube is found by rounding the working color space coordinates of $\mathbf{p}_t$ down to the next working color space grid vertex. The cube that starts at the computed vertex and extends in the positive direction for each coordinate is then the one we are looking for. Now the backtransform is simply a matter of trilinear interpolation. This amounts to an operation in $\mathcal{O}(1)$ per pixel, meaning an image with n pixels can be backtransformed in $\mathcal{O}(n)$.

4. GAMUT MAPPING

The central problem of modern image reproduction is gamut mapping. Thereby gamut means the set of colors reproducible by a device, or process, respectively. Similarly, image gamut stands for the set of colors given by an image. Nowadays it is not unusual to consider a color space like sRGB as input gamut. In image reproduction, gamut mapping means the problem that the output gamut (typically a CMYK color space) is often much smaller than the input gamut. As a consequence, a specified input color has, in general, to be mapped to a reproducible different output color. Of course, the problem increases with growing disparity between input and output gamut.

Since ICC-CMS normally considers color spaces like sRGB or ECI RGB as input gamut, the ICC-CMS is always in a worst case situation. On the other hand, an image as a set of colors defines only a subset of the underlying color space as input gamut. Obviously, this behavior induces the chance for better gamut mapping algorithms, so-called image-dependent gamut mapping strategies. However, the situation is not as clear as it may appear to be at first glance.

Normally, a gamut is considered a 3D polyhedron and gamut mapping is heavily related to its geometric shape. However, in the case of image gamuts, only a point-cloud of colors is given. Therefore, the gamut surface has to be determined first, which is a non-trivial task. There are several approaches, ranging from convex hull to more complex algorithms like alpha shapes.[7] A solution with good visual performance is the one by Schuberth et al.[489] , but for real-life applications it is too slow.

Here we propose a new gamut mapping approach that works on the image gamut without requiring the explicit determination thereof. A popular gamut mapping strategy is the mapping along a ray in the direction of a focus point, see figure 2. The ray is defined by the color C being mapped and the focus point $\mathbf{p}_{focus}$. The colors C_i and C_o on the input and output gamut surfaces $\mathcal{G}_i$ and $\mathcal{G}_o$ are defined by the intersection of the mapping ray with the respective surface.

Then the mapping is defined as a function of the distances R, R_i and R_o of the colors C, C_i and C_o to the focus point. Usually, $R < R_i$ and $R_o < R_i$ holds, meaning an input color lies inside the input gamut and the output gamut is smaller than the input gamut. If this is not the case, the configuration is degenerated and R is either clipped to R_o (mapped radius $R_m = R_o$) or left alone ($R_m = R$), depending on the relative position of C to C_o. Otherwise, the amount of compression should be calculated from the relative position of C to C_i and scaled with R_o, for instance as

$$R_m = \left(1 - \left(1 - \frac{R}{R_i}\right)^{\alpha}\right) \cdot R_o \tag{1}$$

where R_m stands for the radius of the resulting output color and α is the degree of non-linearity. R is given by C. R_o is the radius of the block representative for the containing block of C and can be retrieved in $\mathcal{O}(1)$ from the block structure introduced in section 3.1.1. But the image gamut radius R_i is unknown and has to be estimated first. This can be done using standard methods like the interpolation from all points with similar color to C (see figure 3). Technically this interpolation is realizable by a filter technique. For that purpose we use a Gauss filter that operates on color distances and scales these with the standard deviation σ_c:

$$G_{\sigma_c}(\mathbf{c}) = e^{-(\Delta L^2 + \Delta a^2 + \Delta b^2)/(2\,\sigma_c^2)}$$

For visual and efficiency reasons, a confinement of the interpolation to the local pixel neighborhood (see figure 4) suffices, which results in the application of another Gauss filter, but this time operating on pixel distances, scaled with standard deviation σ_s:

$$G_{\sigma_s}(\mathbf{x}) = e^{-(\Delta x^2 + \Delta y^2)/(2\,\sigma_s^2)}$$

Since the sequential application of two Gauss filters is equivalent to the application of the combination of the two, we arrive at the following overall solution:

$$G_{\sigma_s,\sigma_c}(\mathbf{x},\mathbf{c}) = e^{-\frac{1}{2}\left((\Delta x^2 + \Delta y^2)/\sigma_s^2 + (\Delta L^2 + \Delta a^2 + \Delta b^2)/\sigma_c^2\right)} \tag{2}$$

This is, in fact, an edge-preserving smoothing filter, which has the pleasant side-effect of preserving details without introducing halo-artifacts. Our implementation follows Zolliker,[10] which is an efficient randomized approach. Once R_i is determined, the compression of C along the mapping ray is performed.

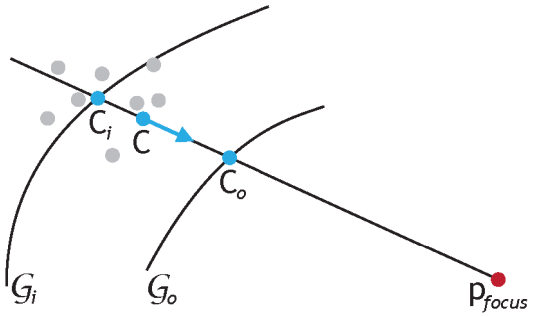

Figure 2. Gamut Mapping Strategy: Mapping of color C along a mapping ray in the direction of the focus point.

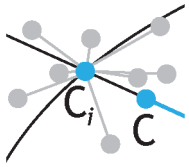

Figure 3. Color Distance Filter: C_i is the color to be estimated, the gray dots represent the points with similar colors to C from which to interpolate.

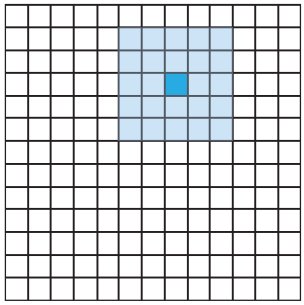

Figure 4. Pixel Distance Filter: Only the local pixel neighborhood is taken into account. The grid represents an image consisting of a number of pixels. The blue pixel is the one being mapped and the light blue area is its local pixel neighborhood.

4.1 Gray Axis Transform

Since the gray axes of the input and output gamuts generally do not coincide, it is important to apply a correction, such that gray will be reproduced as gray and not blue-grayish or yellow-grayish. As a simple solution, a correction vector $\mathbf{v}_{\text{white}}$ from input white to output white and a correction vector $\mathbf{v}_{\text{black}}$ from input black to output black are calculated and then used for the transform. The input white and black are given by the two intersections of the device gamut with the sRGB gray axis. The white intersection is simply the block representative of the block that contains sRGB white, and analogous for black. These vectors are calculated only once, in the initialization. Then for every pixel within a certain region of the gray axis, a linearly interpolated correction vector is added based on the pixel's L-value.

$$\mathbf{v}_{\text{white}} = \mathbf{p}_{\text{w,out}} - \mathbf{p}_{\text{w,in}} \quad , \quad \mathbf{v}_{\text{black}} = \mathbf{p}_{\text{b,out}} - \mathbf{p}_{\text{b,in}}$$

$$\mathbf{p} = \mathbf{p} + \gamma(d) \cdot (t \cdot \mathbf{v}_{\text{white}} + (1-t) \cdot \mathbf{v}_{\text{black}})$$

where t is the interpolation parameter:

$$t = \frac{L_{\text{p}} - L_{\text{b,in}}}{L_{\text{w,in}} - L_{\text{b,in}}}$$

and $\gamma(d)$ the influence of the correction, based on the color distance d of the current color to the gray axis, which is modeled by

$$\gamma(d) = \left(e^{d^2/(2\,\sigma^2)} - d_{\text{off}}\right) \cdot \frac{1}{1 - d_{\text{off}}}$$

The standard deviation σ^2 in effect determines the size of the influence region. In order to guarantee a smooth transition, the distance d_{off} between the lowest reached point of the exponential function and zero is subtracted and the result subsequently scaled to the range [0,1].

4.2 Algorithm

Given an original input image I, a mapped output image I_m is obtained by performing a gamut mapping $\mathcal{GM}(I) = I_m$ as follows:

For every pixel of I:

1. Transform pixel's RGB color C to polar coordinates (R, θ, ϕ) of the working color space, where R stands for the radius and θ, ϕ are the corresponding angles.
2. Find the containing segment of C using the polar angles θ, ϕ and get the corresponding block representative's color radius R_o.
3. Approximate the radius R_i of the input gamut on the mapping ray using the edge-preserving smoothing filter introduced in formula (2). This amounts to calculating a weighted sum of the pixel's radius and its neighbors' radii, where the weights are given by the filter.
4. For $R < R_i$ and $R_o < R_i$: determine mapped radius R_m according to the compression function (1). The degenerated cases are to be clipped to R_o or R respectively, as described above.
5. Convert mapped color $C_m = (R_m, \theta, \phi)$ to cartesian coordinates and do gray axis transform (section 4.1).
6. If C_m is out-of-gamut, clip to block representative.
7. Store C_m in mapped image I_m.

4.3 Parameters

The behavior of our algorithm can be controlled by the following parameters. The default values were determined by visual examination.

Working color space. In the prototype either CIELAB or IPT.

Color standard deviation. The range of colors relative to each pixel that is having an influence is defaulted to $40\Delta E$. This amounts to a standard deviation of the filter's color part of $15\Delta E$.

Filter size / Distance standard deviation. The standard deviation of the filter's distance part is defaulted to 5, which delivers a good quality vs. time trade-off. This amounts to a filter size of 15×15.

Pixel percentage. 10% of the filter pixels are actually taken into account. Bigger percentages dramatically increase the computational complexity, while not producing perceivably superior results. This only works because we systematically distribute the probability that a given pixel gets chosen to a Gaussian normal distribution with the midpoint at the center pixel of the filter.

Non-linearity exponent. A non-linearity exponent of 1.4 for the compression curve preserves saturation but also does not over-compress the saturated colors.

Influence of gray axis transform. The influence region of the gray axis transform should be pretty large, about $100\Delta E$.

5. RESULTS

5.1 Computational Performance

5.1.1 Fitting the measurement data to the WCS grid

The construction of the interpolation grid for the backtransform requires the sampling of the working color space in predefined steps in a 3D region that completely covers the device gamut. Depending on the sampling step, this can take a while (see table 1). However, this is acceptable because the fitting is done offline.

Grid step [ΔE]	Time [s]
10	0.08
5	0.5
3	2.5
2	8.2
1	60.81

Table 1. Performance of building WCS grid

5.1.2 Backtransform

The backtransform is fast, it averages at about $1.13\mu s = 1.13 \cdot 10^{-6} s$ per pixel in the current implementation, which means about 890'000 pixels can be backtransformed per second (c.f. table 2).

Dimension	Number of Pixels	Time [s]	Time per Pixel [μs]	Pixels per Second
320x400	128'000	0.15	1.17	853'333
760x760	577'600	0.66	1.14	875'151
1024x1024	1'048'576	1.20	1.14	873'813
1500x1500	2'250'000	2.38	1.06	945'378
1654x2339	3'868'706	4.45	1.15	869'372

Table 2. Performance of backtransform

5.1.3 Gamut Mapping

Gamut mapping is clearly the bottleneck of the application: since it is filter-based, for every pixel being mapped also a certain amount of its neighborhood is taken into account. Assuming the mapping was done with the default parameters described in section 4.3, it averages at about $11.14\mu s = 11.14 \cdot 10^{-6} s$ per pixel, meaning about 90'000 pixels can be mapped per second (c.f. table 3).

Dimension	Number of Pixels	Time [s]	Time per Pixel [μs]	Pixels per Second
320x400	128'000	1.40	10.93	91'428
760x760	577'600	6.35	10.99	90'960
1024x1024	1'048'576	12.18	11.61	86'089
1500x1500	2'250'000	25.76	11.44	87'344
1654x2339	3'868'706	41.57	10.75	93'064

Table 3. Performance of gamut mapping with default settings

5.2 Quality

5.2.1 Accuracy of the Backtransform

In order to confirm the accuracy of our concept, we generated different RGB charts, printed and measured them with the eye-one photospectrometer. Subsequently, we compared the original values with the measured values.

We generated RGB charts

- with the original samples,
- with the colors half between the originals
- and with random samples.

We also varied the accuracy of the backtransform grid. As you can see in table 4, the grid accuracy has a high influence on the result.

RGB chart	Grid step [ΔE]	ΔL	Δa	Δb	ΔE
orig	10	1.37730	0.08768	1.13970	1.78985
orig	2	0.64252	0.05293	0.28944	0.70669
orig	1	0.83013	0.10791	0.53272	0.99225
half	1	0.89489	1.50364	0.45060	1.80687
random	1	0.43078	2.01268	2.46780	3.21349

Table 4. Average errors with working color space CIELAB

5.2.2 Visual Performance of the Gamut Mapping Algorithm

To confirm the visual quality of our gamut mapping algorithm, we did a psycho-visual experiment in collaboration with Iris Sprow and Zofia Baranczuk. Five different algorithms were compared: Two of the standard reference algorithms (*HPminDE* and *SGCK*) recommended in the CIE technical report TC 8-03,[11] one image-dependent algorithm by Kolas and Farup,[3] one image-dependent algorithm by Schuberth et al.[4] and our algorithm.

A set of a hundred images was mapped to the ISO newspaper gamut with each of the algorithms, resulting in a test set of 500 images. These were subsequently compared by 28 observers by pair comparison to the original. The data was evaluated using Thurstone's scaling method.

Our algorithm performed very well, see figure 5. It beat both the standard algorithms and Kolas, but was not quite as good as Schuberth. For the authors, the comparison with Schuberth is of particular importance.

On the one hand because this algorithm also works on the image gamut and on the other hand produces very good visual quality. The disadvantage is the very high running time (approx. 30 minutes per image). Our approach was planned as an efficient alternative with comparable visual quality. We achieve a significantly better computational performance - our running time is in the order of several seconds, depending on image size. The visual quality is in a comparable range, but does not quite reach Schuberth's in most cases.

This evaluation is to be understood as a pilot study. There is definitively potential for improvement, in particular in the context of filter design and parameter optimization. Since the algorithm is highly parametrized it is a big challenge to determine the optimal configuration. It can only be figured out with extensive testing and analysis, which was outside the scope of this paper. But the potential is there and will be addressed in future work.

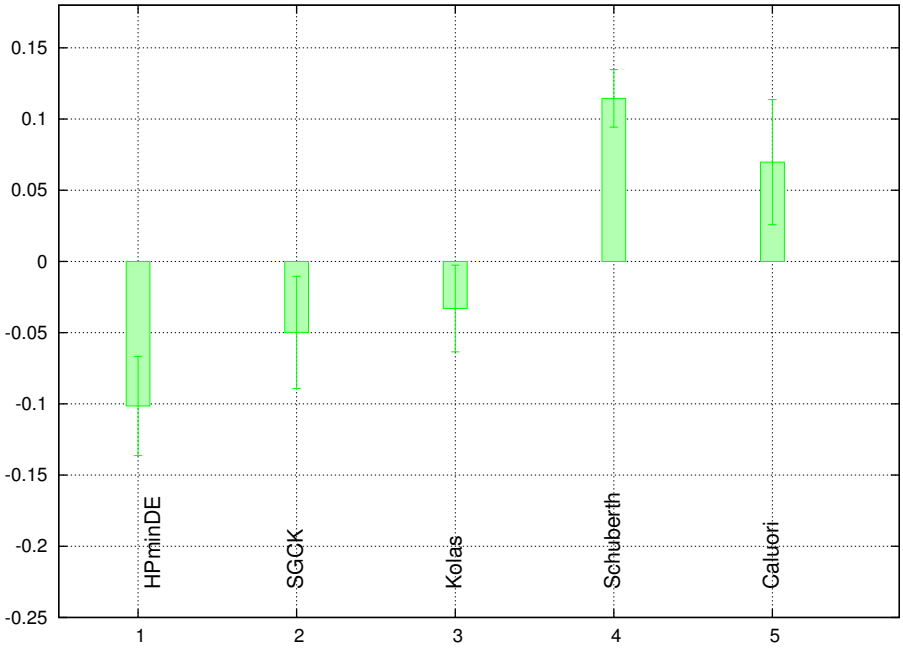

Figure 5. Results of psycho-visual test

6. CONCLUSION

In this paper we demonstrated that there are practical alternative color management systems with better visual performance. Avoiding the drawbacks of ICC-CMS as a distributed software standard, we were able to combine a sophisticated gamut mapping algorithm with a simple interpolation technique for the color transform. The visual advantages were verified in psycho-visual experiments. As a reproduction system we reach proofing accuracy. As an integral RGB-approach our concept is user-friendly and less susceptible to maloperation by the user.

In the future, our prototype will be improved in a number of ways. For instance, by choosing a more adapted filter technology, optimization of our algorithmic parameters and a more efficient implementation. Also, the question of the best performing working color space is open. We will concentrate on the trade-off between running time and image quality.

ACKNOWLEDGMENTS

In particular we would like to thank Iris Sprow and Zofia Baranczuk for the psycho-visual gamut mapping test data. Furthermore, we would like to extend our thanks to all co-workers of the media technology lab at EMPA and everyone who participated in the psycho-visual experiment.

REFERENCES

[1] Morovič, J., [*Color Gamut Mapping*], Wiley, West Sussex, England (2008).
[2] Adobe, [*PDF-Reference*], Peachpit Press, Berkeley, CA, fifth ed. (2005).
[3] Kolas, O. and Farup, I., "Efficient hue-preserving and edge-preserving spatial color gamut mapping," in [*Proc. 15th Color Imaging Conference Final Program and Proceedings*], 207–209 (2007).
[4] Giesen, J., Schuberth, E., Simon, K., Zolliker, P., and Zweifel, O., "Image-dependent gamut mapping as optimization problem," *IEEE Transactions on Image Processing* **16**, 2401–2410 (Oct 2007).
[5] Kimmel, R., Shaked, D., Elad, M., and Sobel, I., "Space-dependent color gamut mapping: A variational approach," *IEEE Transactions on Image Processing* **14**, 796–803 (Jun 2005).
[6] Morovič, J., "Gamut mapping," in [*Digital Color Imaging*], Sharma, G., ed., 652ff, CRC Press (2003).
[7] Cholewo, T. J. and Love, S., "Gamut boundary determination using alpha-shapes," in [*Proc. 7th IS&T/SID Color Imaging Conference*], (1999).
[8] Giesen, J., Schuberth, E., Simon, K., and Zolliker, P., "A kernel approach to gamut boundary determination," in [*Proc. 14th European Signal Processing Conference*], (2006).
[9] Giesen, J., Schuberth, E., Simon, K., and Zolliker, P., "Toward image-dependent gamut mapping: Fast and accurate gamut boundary determination," in [*Proc. 17th IS&T/SPIE Symposium on Electronic Imaging*], (2005).
[10] Zolliker, P. and Simon, K., "Retaining local image information in gamut mapping algorithms," *IEEE Transactions on Image Processing* **16**, 664–672 (Mar 2007).
[11] CIE, "Guidelines for the evaluation of gamut mapping algorithms," Tech. Rep. TC 8-03, CIE (2003).
[12] Sharma, G., [*Digital Color Imaging*], CRC Press, Boca Raton, FL (2003).
[13] Sharma, A., [*Understanding Color Management*], Delmar, Clifton Park, NY (2004).
[14] Green, P. and MacDonald, L., [*Colour Engineering*], Wiley, West Sussex, England (2002).
[15] Green, P., [*Understanding Digital Color*], Graphic Arts Technical Foundation, second ed. (1999).
[16] Giorgianni, E. J. and Madden, T. E., [*Digital Color Imaging*], Addison-Wesley, One Jacob Way, MA (1998).
[17] Morovič, J. and Luo, M. R., "The fundamentals of gamut mapping: A survey," *Journal of Imaging Science and Technology* **45**(3), 283–290 (2001).
[18] Ebner, F. and Fairchild, M., "Development and testing of a color space (ipt) with improved hue uniformity," in [*Proc. 6th IS&T/SID Color Imaging Conference*], 8–13 (1998).
[19] Caluori, U., *An Alternative to Color Management Systems*, Master's thesis, ETH Zurich (2007).