Visualization and Interactive Manipulation of Color Gamuts

Ivar Farup, Jon Y. Hardeberg, Arne M. Bakke,
Ståle Kopperud, and Anders Rindal
Gjøvik University College
Gjøvik, Norway

Abstract

Several tools and techniques for the visualization of color gamuts have been presented in the past. We present a short survey on the topic, and conclude that tools with the possibility for interactive color adjustment in some color space are almost absent. Therefore, a new tool which combines the known techniques with the possibility of interactive gamut mapping is presented along with suggestions for future work. The motivation for developing the new tool is threefold: First, it will serve as an important pedagogical tool in the teaching of color engineering. Secondly, we believe that the tool will prove helpful in research related to color reproduction. Finally, we hope that the tool can be used in the production of high quality color images in the future.

1. Introduction

Color gamut mapping is an integral and important part of color management, and several gamut mapping algorithms have been proposed.[1] Most of the algorithms are formulated as geometric deformations of a color gamut – either the image's source or the image gamut directly – in some device independent color space. The transformations involved are often complex, and it is not straightforward to imagine how they influence the gamut and the resulting color image. In order to achieve a better understanding of the mechanisms involved, different methods for visualizing color gamuts have been developed in the past.

A short survey on existing techniques and tools is presented in Section 2. In Section 3, a new tool for visualization of image and device gamuts and interactive manipulation of image gamuts (i.e., interactive color gamut mapping) is presented. The emphasis will be on implementation rather than application of the new tool. The tool combines many of the techniques which are found in the literature. Finally, some concluding remarks are drawn and suggestions for future work outlined.

2. Color Gamut Visualization

2.1. Visualization Strategies

Since color in tristimulus colorimetry can be represented by three values, color gamuts can be represented by objects in a 3D color space. Hence, 2D visualizations of gamuts must reduce the dimension either by 2D rendering of a 3D scene, or by simplifying the problem otherwise, e.g., by only showing certain cross sections.

The traditional way of visualizing a device gamut is to plot its outline in a 2D xy chromaticity diagram.[2] Alternative variants include plotting the gamut boundary in CIELAB as L^* vs. C^*_{ab} for given values of h_{ab},[3] or viewing projections of the gamut solid onto different principal planes, e.g., (a^*, b^*), in CIELAB.[4–6] Herzog[7] plotted C^*_{ab} as a function of L^* at constant hue along with a histogram (as circles) of an image, thus indicating out-of-gamut colors. Other instructive techniques include plotting gamut volume versus h_{ab}, L^*, and C^*_{ab},[5] or showing C^*_{ab} in grayscale as a function of h_{ab} and L^*.[3]

Already in 1935, MacAdam introduced the third dimension of the xyY chromaticity diagram by the use of contour lines.[8] Similar techniques have also been used later.[5] MacAdam also presented a stereoscopic photograph of a surface bounding all attainable colors in illuminant C in the xyY color space.[8] In 1988, Stone et al. presented orthographic views of idealized device color gamuts as "true" colored solids.[9] Following the increase in computing power, quasi-realistic perspective renderings of color gamuts represented as colored solids,[2,10,11] wireframe outlines,[6,12] or combinations of both,[5,13] have become common. Particularly appealing is the use of partially transparent solids.[14,15] When the gamut to be visualized is known from discrete measured colors or is an image gamut, it can also be visualized as a point cloud.[6,16] Another instructive technique is to depict the gamut as a leaf structure consisting of "true" colored planes of, e.g., constant hue or constant lightness, in a 3D scene.[4] Outline of the gamut boundary in such planes have also been used for visualization more recently.[17] A slightly different approach is to work directly in cylindrical coordinates, and plot the chroma as a function of lightness and hue, $C^*_{ab}(L^*, h_{ab})$, as a surface.[3,5–7,18]

Comparison of different color gamuts can be done by representing the gamuts as objects of different types (solid, wire-frame, etc.) in the same coordinate system,[6,11,14,15] or more directly by generating a solid from the difference between device gamuts.[10] These differences can become even clearer when only parts of the gamuts are shown, either as solids[15] or as color coded point clouds.[16]

Other quantities related to color can also gain from visualization. Displacement vectors associated with a gamut mapping algorithm can be shown as arrows or line segments,[2] errors associated with measurement uncertainty as spheres,[2] and color histograms for images as differently sized spheres.[19]

2.2. Gamut Boundary Determination

The determination of the color gamut boundary plays a crucial role in color gamut visualization. This can be performed in at least two principally different ways. First, it can be found from the measured data points (colors in some color space). Secondly, it can be derived from an analytic model. Some of the methods among the former set are directly applicable to other color sets such as images, whereas methods of the latter category necessarily depends strongly on the device.

2.2.1. Point Based Methods

The simplest way of obtaining the device gamut from measured colors, is to assume that the gamut boundary is preserved between device dependent and device independent color spaces. Stone et al.[9] measured the most extreme colors (gamut corners) of a printing system and connected them with planes. Bolte[10] obtained gamuts by direct triangulation of the measured printed colors laid out in a regular structure in the device color space. An improvement of this technique is to check for well-behavedness, i.e., that the gamut surface in the device color space corresponds to the surface in the device independent color space, and that no tetrahedra are mirrored.[20,21]

If there is no imposed structure in the measured data, a convex hull algorithm can be applied directly, resulting in a purely convex gamut approximation.[14,15] This strategy has recently been applied to gamuts derived from ICC profiles.[6] Guyler[12] recently argued that the data tend to be more convex in the CIEXYZ color space, and that the convex hull algorithm thus should be performed there and later transformed to the desired color space such as CIELAB. A similar solution which works for reasonably well-behaved printers is to compute the convex hull in a linearized dye density space.[22] Balasubramanian and Dalal[23] introduced a method for making the measured data more convex by means of a non-linear function before computing its convex hull. This method seems better suited to printer-like gamuts than to images since it requires the precise selection of an interior point.

From measured colors, the maximum chroma for cells in the (L^*, h_{ab}) plane can be found. These points can have an imposed regular structure, and can thus easily be plotted as a triangulated surface.[3,5] The segment maxima method is a similar method using polar instead of cylindrical coordinates.[24] It consists in subdividing the CIELAB color space into segments of uniform spherical angles (polar and azimuth). The center of the spherical coordinate system is taken either as the point having (L^*, a^*, b^*) coordinates $(50, 0, 0)$ or as the center of mass of the given points. For each segment, the point with the larger radius is stored, and together these points define the gamut boundary. Triangulation of the surface is straightforward due to the imposed regularity. This technique has also been used for image gamut volume determination, using the mass center as the center of the coordinate system.[25]

Recently, Cholewo and Love[13] introduced the use of alpha shapes[26] for gamut boundary determination of both images and devices. An alpha shape is a generalization of a convex hull applicable also to non-convex solids. It is obtained by performing a 3D Delaunay triangulation of the point set and subsequently removing the simplices, i.e., tetrahedra, triangles, and edges, for which there exists a sphere of radius α which does not contain any other points from the set. Cholewo and Love advised that the value of α is chosen at least large enough to make the resulting body consist in a single component. However, they also concluded that the optimum α value is best determined interactively by inspection.[13]

2.2.2. Model Based Methods

An early attempt at deriving the gamut from a physical model was to assume box-shaped reflectance of colorants and calculate CIEXYZ tristimulus values and thereby the gamut.[8] Meyer et al.[14,15] used the Kubelka–Munk[27] equations for determining the gamut of printing systems. The Neugebauer[28] equations were used by Mahy[29] for calculating the gamut of n-ink printing systems.

Inui[17] presented a novel four-step algorithm for computing printer color gamuts based on the assumed one-to-one correspondence between dye amount space and color space.

Herzog developed the concept of gamulyts – an analytical mathematical description of color gamuts – by modeling the gamut as a deformed cube.[7,18] The deformation was modeled by a set of *distortion functions*. He also discussed how this approach can be extended to systems with four or more colorants.

2.3. Interactivity

GamOpt by Kalra[16] is the only software tool known to the authors capable of interactive manipulation of gamut data. The interactivity consists in deformation of the unit circle in (a^*, b^*) with relative scaling of other colors, linear transformation of the L^* axis, linear transformation of the entire gamut, user specified functions for the whole gamut, and automatic homogenizing of gamut points.

2.4. Implementation

A broad range of development tools from general purpose programming languages to high level simulation systems have been used for gamut visualization. GamOpt[16] was implemented in C[30] using the Motif[31] and X11R5[32] libraries on a Unix platform, Matlab[33] and IDL[34] were used by Braun,[3] and the AVS visualization software[35] was used by Rolleston[2] and Meyer.[15] Recently, some modern programming environments like VRML[36] have been used by Morovic,[37] Java3D[38] by Zhao,[6] and JavaScript[39] by Cholewo and Love,[13]. These tools make implementation easier without degrading the performance too much, and thus seem good choices for new systems.

3. ICC3D: A Tool for Visualization and Interactive Manipulation of Color Gamuts

The aim of the authors is to make a visualization software by combining the most instructive techniques found in the literature with novel techniques for interactive altering of color gamuts and individual colors. The new software will thus allow for experimenting with color gamut mapping both for research and educational purposes as well as serving as a real usable tool for optimized color image reproduction.

In order to make the tool as platform independent as possible, the Java 2 SDK[40] was chosen for development. For convenience, the Java3D,[38] Java Advanced Imaging,[41] and Java Media Framework[42] APIs were also used. Unfortunately, these libraries are currently not available on the Macintosh platform, but this will hopefully change in the near future. So far, the current pre-release version of the application has been tested successfully on both Windows and Linux platforms. A screen-shot of the application is shown in Fig. 1. Information on how to obtain a beta version of the software for testing can be found on our website.[43]

Being based upon plug-ins which are loaded dynamically at start-up, the software architecture is extremely modular. The basic functionality of the software is the ability to read color data either as an image in some standard format, as measured colors in the IT8.7/2 file format, as extracted by the gamut tag of an ICC profile, or as color

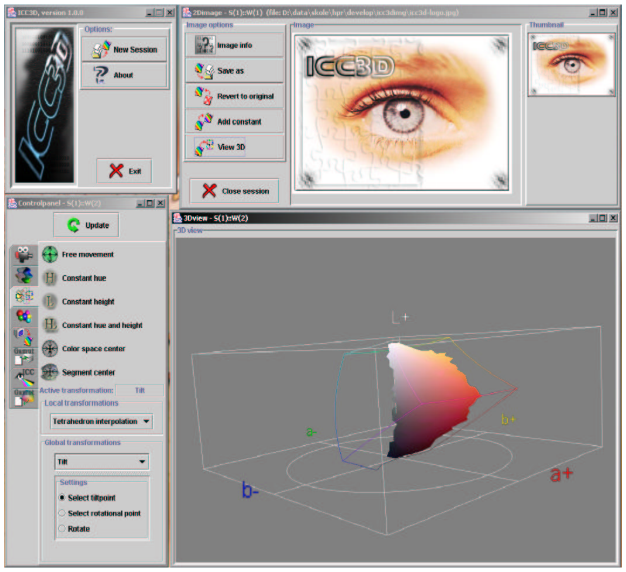

Figure 1: Screen-shot showing ICC3D at work in a Windows environment.

values in a regular file format identical to the one used by Color3D[19] for compatibility. The color data can then be displayed in different ways in various 3D color spaces. The user may interact with the visualization in several ways, possibly under some constraints, in order to change the color data as desired. The plug-ins thus fall into four different categories: color spaces, visualizations, interactions, and constraints.

In practice, a plug-in is implemented as a Java class which extends a given abstract base class. In addition to performing the required work, each plug-in is responsible for creating its own GUI.

3.1. Color Spaces

Images can be loaded in most standard image file formats and viewed on the monitor. So far, both the color data in the input files and the monitor are assumed to be sRGB. The task of each color space plug-in is to convert a given color to a given color space. So far, the sRGB, CIEXYZ, and CIELAB color spaces have been implemented using the well-known analytical transformations. The CIELAB plug-in also provides axes and labels for the visualization.

3.2. Visualizations

The image colors or the color data obtained otherwise can be visualized using either parallel or perspective rendering in the 3D color space provided by the chosen color space plug-in. In the 3D color space, the user can rotate and zoom using the mouse. In addition, several predefined

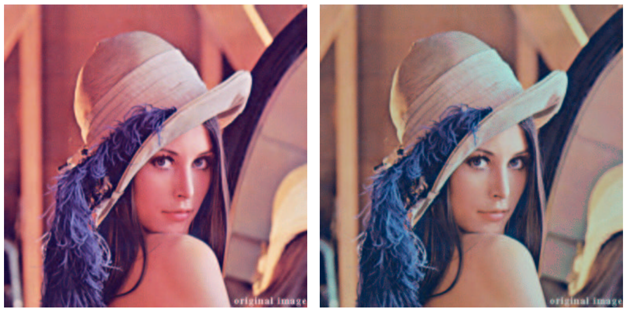

Figure 2: *Sample image before (left) and after (right) the interactive manipulation and gamut mapping shown in Fig. 5.*

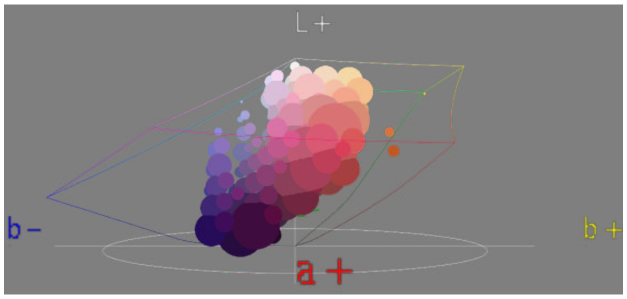

Figure 3: *3D histogram in CIELAB for the image shown in Fig. 2 (left). The deformed cube represents the sRGB gamut boundary.*

views such as straight from above or along the a^* axis with L^* directed upwards, can be chosen by means of push buttons.

Several modes of visualization are implemented. In the simplest mode, every pixel in the image is plotted as a point in the given color space, resulting in a cloud of colored points. With many different colors present in the image, this results in a quite time-consuming rendering process. To cope with this problem, it is possible to quantize the color space into discrete regions and show a pixel for the color space region only if image pixels exist in that region. An extension of this approach is to count the number of pixels in each color space region and plot this as a histogram of differently sized colored spheres. Fig. 3 shows the histogram for the image shown in Fig. 2 (left).

The segment maxima method for determining a triangulation of the gamut surface has also been implemented. The user can select whether $(50, 0, 0)$ or the image mass center should be used as center for the polar coordinate system, and it is also possible to select between different methods for interpolation in the segments where there are no image pixels. The user may choose to relocate the points to the segment center in order to ensure a well-behaved tetrahedra structure. The gamut surface can be visualized as a colored wire-frame, colored solid, partially transparent colored solid, or a combination, cf. Fig. 4.

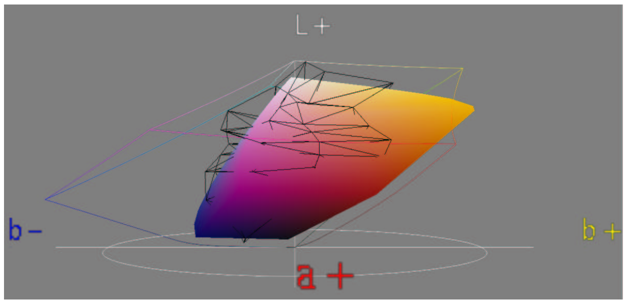

Figure 4: *Segment maxima method applied to the image (wire-frame) in Fig. 2 (left) along with a printer gamut visualized as a convex hull (solid).*

Computation of the convex hull, i.e. the smallest convex volume containing all points, of the given color coordinates has been implemented as another method for deriving the gamut boundary. For this purpose, the QuickHull[44] algorithm was chosen and slightly optimized for the present purpose.

The most advanced visualization method employs the use of alpha shapes[26] for gamut boundary determination. First, the 3D Delaunay triangulation of the colors (preferably quantized) is computed by means of the QuickHull algorithm performed in 4D. The α value (sphere radius) can be set by the user by means of a slider, and the alpha shape is then obtained by removing the simplices corresponding to the α value as described above. The volume of both the initial convex hull and the alpha shape is displayed. An example of an alpha shape is shown in the screen-shot, Fig. 1.

Several device gamuts can be shown together with an image gamut. An example of this is given in Fig. 4.

3.3. Interactions

An important strength of ICC3D is the possibility for interactive change of image colors in the 3D color space. ICC3D operates with two principally different kinds of interactions: local interactions which act on a single color or a single group of colors, and global interactions which operate on the entire image or device gamut.

3.3.1. *Local interactions*

When the image is visualized as a point cloud in color space, individual pixels can be moved using the mouse. The image preview is updated after each move. In the quantized view, it is possible to drag the objects representing collections of pixels. All colors represented by the object can be moved an equal distance in color space. However, since this inevitably leads to banding effects, a better solution can be chosen by the user: A tetrahedra structure

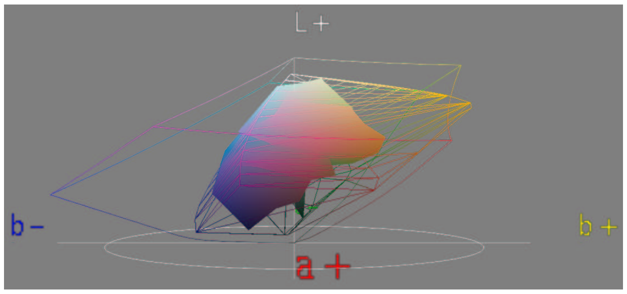

Figure 5: *The image gamut from Fig. 4 (now as solid) interactively manipulated to fit within the device gamut from the same figure (now as wire-frame).*

is built for the entire color space using the histogram points as nodes. This is straightforward since the histogram cells have been chosen as a regular structure. Tetrahedral interpolation is then used for all pixels not lying exactly at the nodal positions.

Although the histogram strategy reduces the number of points somewhat, there are still far to many points to move in order to obtain a reasonable gamut mapping. To cope with this, interactivity with the nodal points of the segment maxima surface is introduced. An interactively altered image gamut is shown in Fig. 5. Upon movement of a boundary point, all of the interior points are updated according one out of two schemes. The simplest solution consists in scaling the translation vector of the boundary point linearly with the radius for all pixels in the original segment, again resulting in banding. The improved solution is to use tetrahedral interpolation, the tetrahedra now consisting of the surface triangles and the central point. For pixels lying outside the segment maxima surface (they exist), extrapolation using the same equations as for the tetrahedral interpolation is applied after selecting the nearest tetrahedron. The resulting image is shown in Fig. 2 (right).

For the convex hull and alpha shapes visualizations, no local interaction is available yet.

3.3.2. Global interactions

Using the global interactions, the user can perform actions which influence all colors, similar to the effects obtainable by ordinary image manipulation programs. The global interactions include moving the gamut in the (a^*, b^*) plane (hue), scaling the L^* axis (contrast), moving along the L^* axis (brightness), radial expansion/compression (saturation), and tilting of the entire gamut (white-point).

3.4. Constraints

Moving points in a 3D color space through a 2D user interface can be quite challenging. To help the user, it is possible to constrain the movement to certain planes or certain lines. These can be chosen as lines toward the nearest point on the gray axis, lines toward color space or gamut center, planes of constant lightness, and planes of constant hue.

4. Conclusion and Future Work

A novel tool for visualization and interactive manipulation of color gamuts has been presented. Although far from complete, it is already at the present stage useful for experimenting with different gamut mapping strategies and for gaining an increased understanding of the mechanisms involved. The tool is based upon a plug-in structure, making it extremely well-suited for future extensions.

For the future, there are several possibilities for adding features which will make tool even more interesting:

- ICC profiles for both input files and monitor in order to increase realism and usefulness. Also, a color appearance model should be included.
- Other forms of constraints when performing interactive color adjustments. For example, it should be impossible to mirror a tetrahedron while moving. This can be solved either by constraining the movement, or by making the movement influence also the movement of neighboring tetrahedra.
- Other algorithms for gamut boundary determination, e.g., convex hulls computed by “convexification” of the image data, cf. the survey above.
- Possibility of selecting regions in the color space and highlighting the corresponding pixels in the image and vice versa. Selections could also be made automatically based on specific properties, e.g., pixels outside given device gamut. It should then also be possible to perform operations (translations etc.) on such groups of points.
- More advanced interpolation functions, e.g., S-curves and clipping.
- Operations on a device gamut as well as on an image gamut, and save the result as a general gamut mapping algorithm in the form of, e.g., an abstract ICC profile.[45]
- Experiment somewhat with the user interface. In the present version, confusingly many parameters are to be controlled by the use of a single mouse. Alternatives include the use of track balls, joy sticks or simply the keyboard.

5. Acknowledgments

The authors would like to thank Mr. Rune Lossius and the library at Gjøvik University College for assisting the literature survey.

6. References

1. Ján Morovič and M. Ronnier Luo. The fundamentals of gamut mapping: A survey. *Journal of Imaging Science and Technology*, **45**(3), 283–290 (2001).
2. Robert Rolleston. Visualization of colorimetric calibration. In *Color Hard Copy and Graphic Arts II*, volume 1912 of *Proc. SPIE*, pp. 299–309 (1993).
3. Gustav J. Braun and Mark D. Fairchild. Techniques for gamut surface definition and visualization. In *Proceedings of IS&T and SID's 5th Color Imaging Conference: Color Science, Systems and Applications*, pp. 147–152, Scottsdale, Arizona (1997).
4. Philip K. Robertson. Visualizing color gamuts: A user interface for the effective use of perceptual color spaces in data displays. *IEEE Computer Graphics & Applications*, **8**(5), 50–64 (1988).
5. Richard L. Reel and Michael A. Penrod. Gamut visualization tools and metrics. In *Proceedings of IS&T and SID's 7th Color Imaging Conference: Color Science, Systems and Applications*, pp. 247–251, Scottsdale, Arizona (1999).
6. Xianfeng Zhao. Implementing an ICC printer profile visualization software. Master's thesis, School of Printing Manamement and Sciences in the College of Imaging Arts and Sciences of the Rochester Institute of Technology (2001).
7. Patrick G. Herzog. Specifying and visualizing colour gamut boundaries. In L. W. MacDonald and M. R. Luo, editors, *Colour Imaging: Vision and Technology*. John Wiley & Sons Ltd. (1998).
8. D. L. MacAdam. Maximum visual efficiency of colored materials. *J. Opt. Soc. Amer.*, **25**, 361–367 (1935).
9. M. C. Stone, W. B. Cowan, and J. C. Beatty. Color gamut mapping and the printing of digital color images. *ACM Transactions on Graphics*, **7**(4), 249–292 (1988).
10. S. B. Bolte. A perspective on non-impact printing in color. In *Color Hardcopy and Graphic Arts*, volume 1670 of *Proc. SPIE*, pp. 1–11 (1992).
11. Peter A. Crean and Robert R. Buckley. Device independent color – who wants it? In *Proc. SPIE*, volume 2171, pp. 267–273 (1994).
12. Karl Guyler. Visualization of expanded printing gamuts using 3-dimensional convex hulls. *American Ink Maker*, **79**(9), 36–56 (2001).
13. Tomasz J. Cholewo and Shaun Love. Gamut boundary determination using alpha-shapes. In *Proceedings of IS&T and SID's 7th Color Imaging Conference: Color Science, Systems and Applications*, pp. 200–204 (1999).
14. Gary W. Meyer, Linda S. Peting, and Ferenc Rakoczi. A color gamut visualization tool. In *Proceedings of IS&T and SID's Color Imaging Conference: Transforms and Transportability of Color*, pp. 197–201, Scottsdale, Arizona (1993).
15. Gary W. Meyer and Chad A. Robertson. A data flow approach to color gamut visualization. In *Proceedings of IS&T and SID's 5th Color Imaging Conference: Color Science, Systems and Applications*, pp. 209–214, Scottsdale, Arizona (1997).
16. Devendra Kalra. GamOpt: A tool for visualization and optimization of gamuts. In *Proc. SPIE*, volume 2171, pp. 297–304 (1994).
17. Masao Inui. Fast algorithm for computing color gamuts. *Color Research and Applications*, **18**(5), 341–348 (1993).
18. Patrick G. Herzog. Analytical color gamut representations. *Journal of Imaging Science and Technology*, **40**, 516–521 (1996).
19. Gabriel Marcu and Satoshi Abe. Three-dimensional histogram visualization in different color spaces and applications. *Journal of Electronic Imaging*, **4**(4), 330–346 (1995).
20. Jon Y. Hardeberg and Francis Schmitt. Color printer characterization using a computational geometry approach. In *Proceedings of IS&T and SID's 5th Color Imaging Conference: Color Science, Systems and Applications*, pp. 96–99, Scottsdale, Arizona (1997). Also in R. Buckley, ed., *Recent Progress in Color Management and Communications*, IS&T, pages 88-91, 1998.
21. Jon Y. Hardeberg. *Acquisition and Reproduction of Color Images: Colorimetric and Multispectral Approaches*. Dissertation.com, Parkland, Florida, USA (2001).
22. J. A. Stephen Viggiano and William J. Hoagland. Colorant selection for six-color lithographic printing. In *Proceedings of IS&T and SID's 6th Color Imaging Conference: Color Science, Systems and Applications*, volume 6, pp. 112–115, Scottsdale, Arizona (1998).
23. Raja Balasubramanian and Edul Dalal. A method for quantifying the color gamut of an output device. In *Color Imaging: Device-Independent Color, Color Hard Copy, and Graphic Arts II*, volume 3018 of *Proc. SPIE*, San Jose, CA (1997).
24. Ján Morovič and M. R. Luo. Gamut mapping algorithms based on psychophysical experiment. In *Proceedings of IS&T and SID's 5th Color Imaging Conference: Color Science, Systems and Applications*, pp. 44–49, Scottsdale, Arizona (1997).
25. Ryoichi Saito and Hiroaki Kotera. Extraction of image gamut surface and calculation of its volume. In *Proceedings of IS&T and SID's 8th Color Imaging Conference: Color Science and Engineering*, pp. 330–334, Scottsdale, Arizona (2000).
26. Herbert Edelsbrunner and Ernst P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, **13**(1), 43–72 (1994).
27. P. G. Engeldrum. Computing color gamuts of ink-jet printing systems. In *SID Proceedings*, volume 27, pp. 25–30 (1986).
28. H. E. J. Neugebauer. Die theoretischen Grundlagen des Mehrfarbenbuchdrucks. *Zeitschrift für wissenschaftliche Photographie, Photophysik und Photochemie*, **36**(4), 73–89 (1937).
29. M. Mahy. Calculation of color gamuts based on the Neugebauer model. *Color Research and Application*, **22**, 365–374 (1997).
30. Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall Inc. (1988).
31. The Open Group. Motif. *http://www.opengroup.org/desktop/motif* (Visited 2002).
32. The X Window System. *http://www.x.org* (Visited 2002).
33. The MathWorks. Matlab. *http://www.mathworks.com* (Visited 2002).
34. Research Systems Inc. IDL: the Interactive Data Language. *http://www.rsinc.com/idl* (Visited 2002).
35. AVS: Advanced Visual Systems. *http://www.avs.com* (Visited 2002).
36. VRML97: The Virtual Reality Modeling Language. International Standard ISO/IEC 14772-1:1997 (1997).
37. Ján Morovič. Gamut mapping. *http://colour.derby.ac.uk/~jan/gamut_mapping.html* (Visited 2002).
38. Sun Micro Systems. Java3D. *http://java.sun.com/products/java-media/3D* (Visited 2002).
39. JavaScript. *http://javascript.internet.com* (Visited 2002).
40. Sun Micro Systems. Java. *http://java.sun.com* (Visited 2002).
41. Sun Micro Systems. Java Advanced Imaging. *http://java.sun.com/products/java-media/jai* (Visited 2002).
42. Sun Micro Systems. Java Media Framework. *http://java.sun.com/products/java-media/jmf* (Visited 2002).
43. The Color Lab at Gjøvik Universtiy College. *http://www.fargelab.no* (Visited 2002).
44. C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software*, **22**(4), 469–483 (1996).
45. International Color Concortium. Specification ICC.1:2001-12. File Format for Color Profiles (Version 4.0.0) (2001).