DERIVATION OF 3-DIMENSIONAL GAMUT DESCRIPTORS FOR GRAPHIC ARTS OUTPUT DEVICES

William Kress and Michael Stevens [1]

Keywords: 3D, Bezier, CIE, Gamut, Gamut Surface, Matlab.

Abstract: Device independent color output requires at least two basic characteristics be known for each system component to produce color rendering that is both accurate and pleasing: (1) relationship between CIE color and the device signals required to produce an in-gamut rendering of that color, and (2) description of the maximum color gamut of the device. Color management programs use these (and other) characteristics to optimize output quality. This paper addresses output gamut determination and describes color space sampling, numerical techniques for summarizing and organizing the sampled gamut and programs for visualizing color gamuts.

Introduction

The accurate description of color gamut is a necessary component for the simulation of graphic arts printing and proofing devices, and the optimization of color output for display. Non-optimal handling of gamut capabilities can reduce image quality, and mismatches can cause image artifacts and color errors. Many techniques for appearance optimization have been proposed, all require that color and neutral gamut be known.

Frequently, color gamuts have been plotted as a projection onto a chromaticity diagram as shown in Figure 1. This obscures the real 3-dimensionality of the gamut, and it is only the 3-dimensional description of the gamut that is useful. Such a view of the gamuts of Figure 1 is shown in Figure 2. It is the intention of this paper to present a method of accurately determining the gamut surface for color printing systems and illustrate a commercially available visualization program for plotting the results.

Past work (Bell, 1993) has described methods of locating the gamut surface using tetrahedral interpolation. The technique described in this paper uses spline interpolation to accomplish the same.

---

[1]William Kress, NewGen Systems Corporation, Fountain Valley, CA. Michael Stevens, CalComp, Inc., Anaheim, CA.

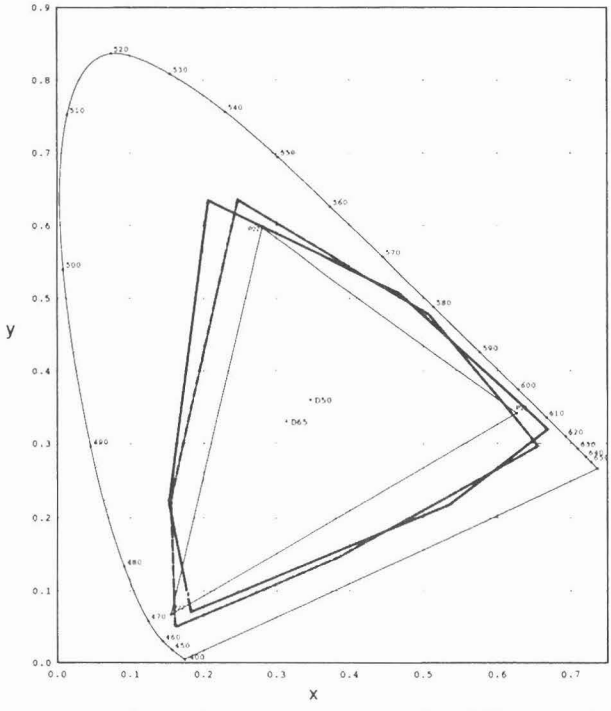

Figure 1. Comparison of two color gamuts for different D2T2 colorants using x, y chromaticity coordinates.

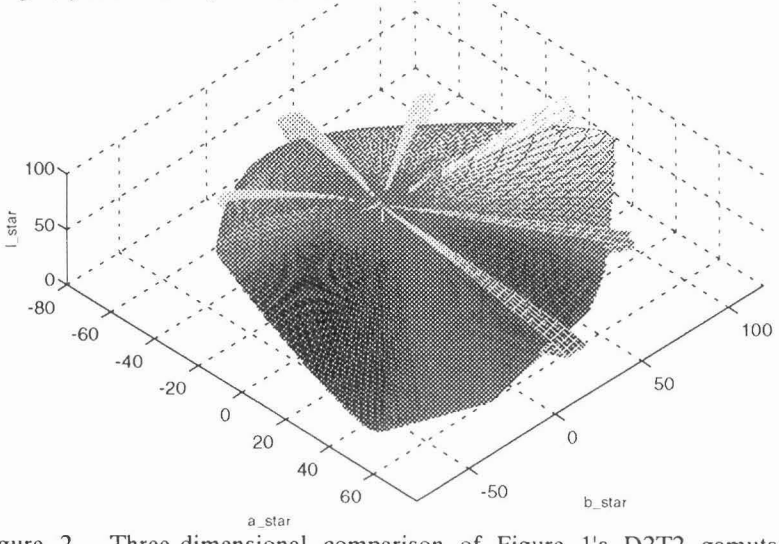

Figure 2. Three-dimensional comparison of Figure 1's D2T2 gamuts. The solid surface shows photographic colorants and the cut surface shows SWOP colorants.

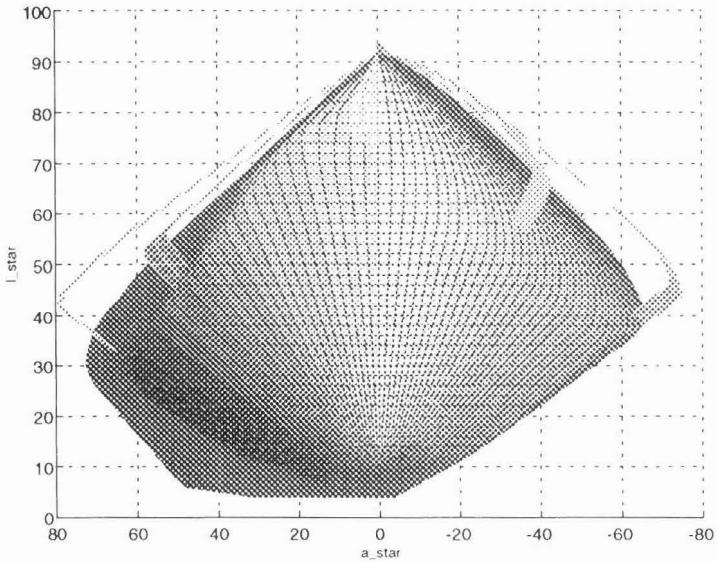

Figure 3. Another viewpoint of the D2T2 gamut shown in Figure 2.

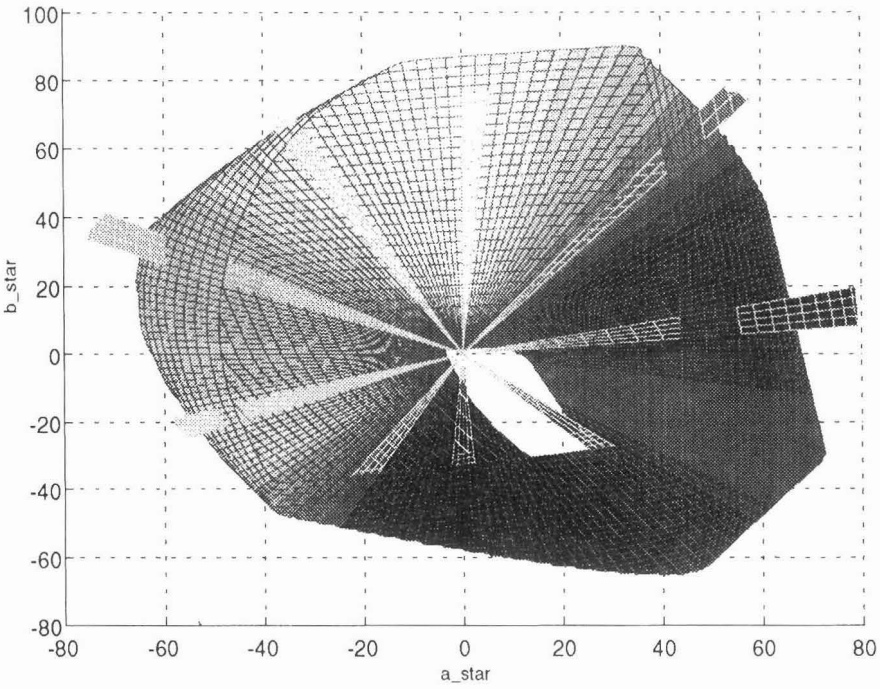

Figure 4. The gamut of Figures 2 and 3 with the L* values greater than 60 set to NaN (see Appendix).

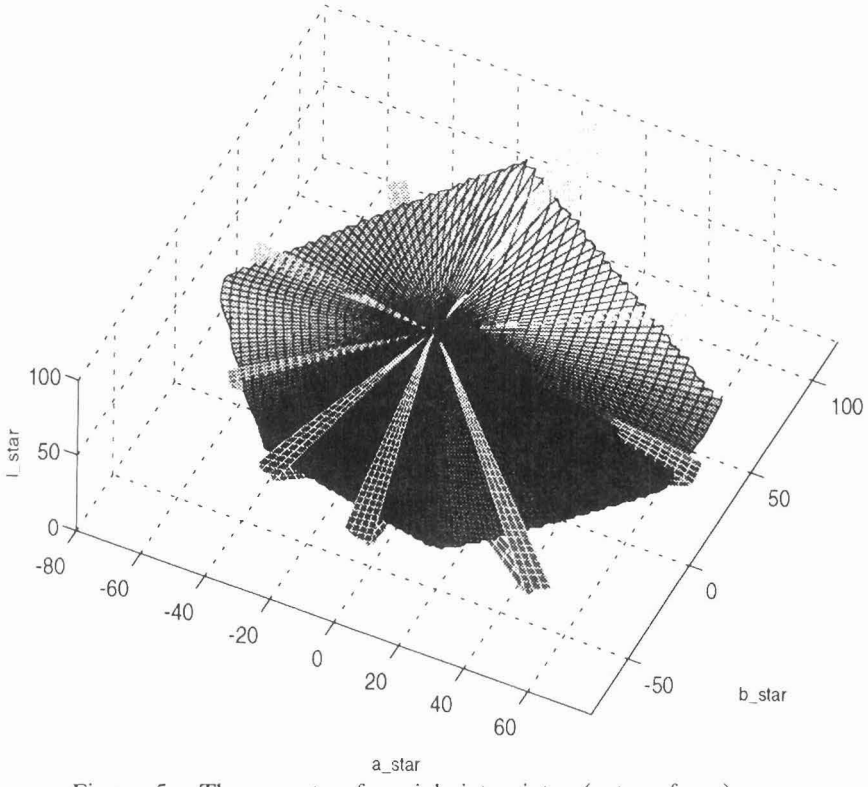

Figure 5. The gamuts of an ink jet printer (cut surfaces) versus a thermal wax printer (solid surface).

Color gamut visualization tools have been developed (Meyer, 1993); because of resource limitations, it was desired that a readily available application program be purchased and used. This report illustrates a technique for identifying data on the gamut surface, and techniques for plotting the data. All Figures shown herein are plotted using Matlab, and the Appendix describes the required data formats, the 3-D plotting routines and presents ideas for coloration and shading to make the gamut surface more meaningful. Figures 3, 4, and 5 illustrate some of Matlab's flexibility in constructing and displaying three-dimensional surfaces.

Gamut Determination

There are two main methods for populating a color gamut descriptor. One, an empirical approach, requires that many measurements be taken from printed patches that are on or inside the gamut surface, and the other, a deterministic or theoretical approach, involves the derivation of a model of the printing system (e.g., Engeldrum, 1986, Berns, 1993). Both have been

tried, and both have advantages and disadvantages.

Theoretical models require an in-depth understanding of all the effects involved with the printing process. By modeling all the forces at work for a given output process, the model can predict the outcome. This type of model can produce accurate results with relatively little data acquisition. The primary disadvantage of such models, is that a new model must be developed for each output process, and the derivation of the construction and parameters for the model is time consuming.

Empirical models do not require any knowledge of the process. They depend on many measurements to ensure accuracy. Empirical models can be applied to any output process. All gamuts described herein were derived by the empirical method.

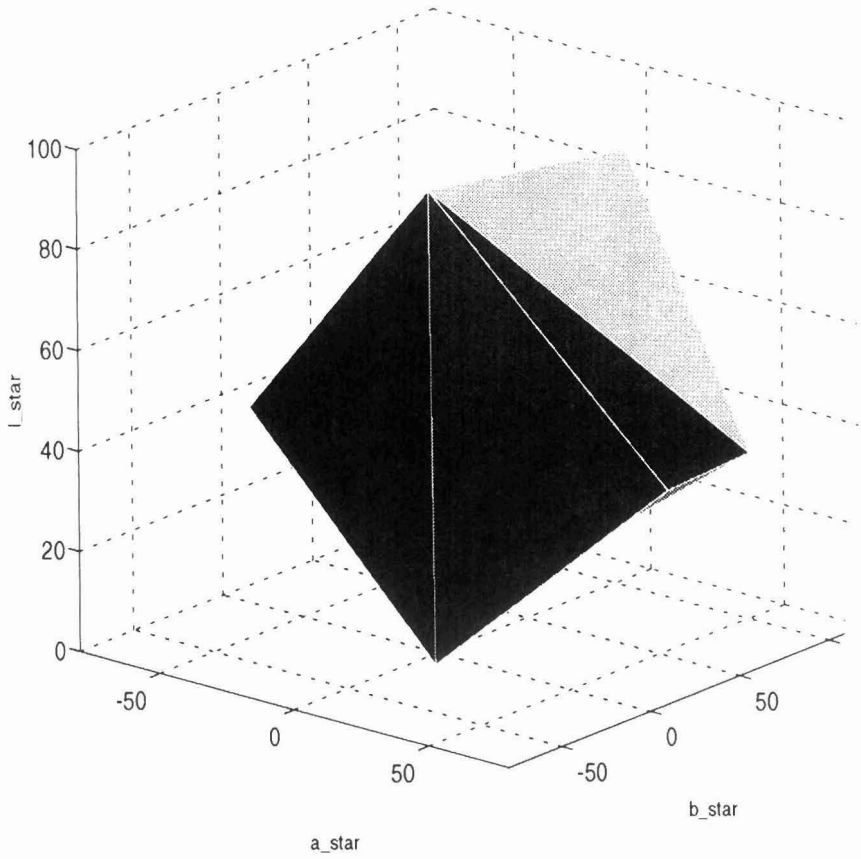

Figure 6. The color gamut of the D2T2 SWOP media shown in previous Figures. Only the primary (CMY), secondary (RGB), paper Dmin and

black were used to describe the color gamut. The gamut was constructed simply by connecting these points.

Algorithms

There are three types of data that can be encountered with gamut descriptors: color space data points, color space data points on the surface of the gamut, and color space data points with corresponding device input signals. The first two data sets can only provide information about the location of the gamut surface, and consequently are only useful when doing gamut comparisons. The third data set not only provides enough information to approximate the gamut surface, but allows us to generate a transfer function that will transform device signals to color space coordinates and the inverse.

Color Space Data Points

If our initial data set is a series of data points that may or may not be on the gamut surface, the task of accurately approximating the gamut surface is a difficult one.

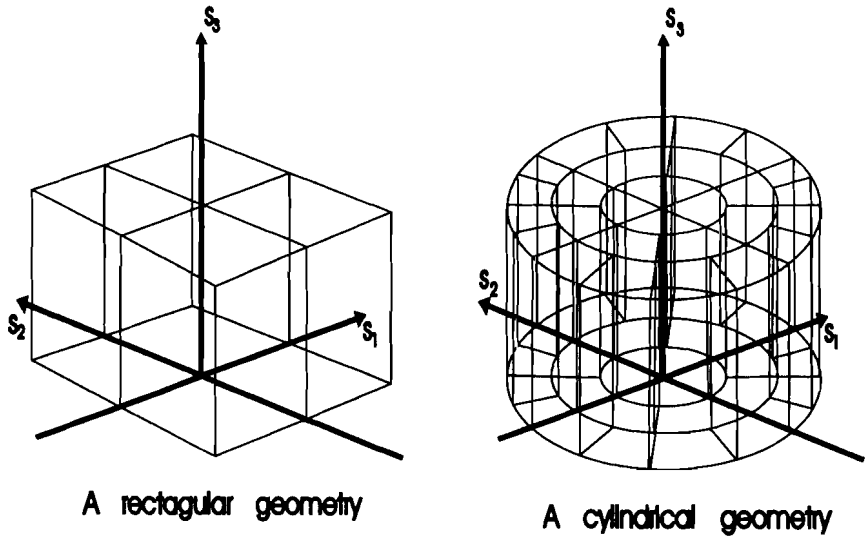

Figure 7. Examples of possible voxelization geometries.

The simplest technique for dealing with this type of data is voxelization. In voxelization the color space is subdivided into discrete voxels (a three-dimensional version of a pixel), using a variety of geometries to suit the users needs. If a data point falls within a voxel, that voxel is considered

solid, otherwise the voxel is considered empty space. This will produce a three dimensional solid model of the shape of the gamut. The only drawback of this technique is the large number of data points needed to produce accurate results. Large voxels will result in aliasing. Therefore, it is highly desirable to use the smallest possible voxel size. There is however a lower limit on the size of the voxels. As the size of the voxels approaches the spacing between data points, the gamut solid will begin to develop holes.

The recommended approach to this problem is to find the convex hull for the particular set of data points. This approach is appealing for two reasons: for every set of data points there exists a unique solution, and there are several published works on algorithms that determine convex hulls. However, this technique has one major shortcoming; gamut surfaces are often concave. Therefore, convex hulls should only be used in the crudest of approximations.

Color Space Data Points On the Gamut Surface

This data set conveys enough information to approximate concave surfaces. Initial attempts to solve this problem revolved around picking a reference point **O** inside the gamut surface. Then an infinitesimally small tetrahedron (referred to as the seed) was constructed around point **O**. As each data point was processed additional tetrahedrons were added to the seed to construct a surface. This technique worked with limited success, and has three flaws: (1) to locate the gamut surface the starting point **O** must be inside the gamut surface, (2) the final result is dependent on the selection of the point **O**, and (3) the algorithm is not capable of approximating all simple closed surfaces.

This lead to the development of the algorithm recommended for the analysis of this data type. The algorithm is as follows:

```
find_convex_hull
sort_data_points
while (unused point exist)
    find_shortest_distance
    add_point_to_surface
    move_point_to_used_set
while ( not fully optimized )
    optimize_surface
```

find_convex_hull - This is a logical starting point of the algorithm. Rather than expand a surface out from a point, this is equivalent to collapsing an infinite surface onto the data points.

sort_data_points - The data points are then sorted into two categories:

primary, and secondary. Primary data points are defined as points that are a part of the convex hull surface. Secondary points are defined as points enclosed inside the convex hull. The secondary points are now further classified into two groups: used, and unused, and all the secondary points start out as unused.

find_shortest_distance - Search through all the combinations of unused points and triangular facets, and determine the shortest distance between the two. The distance is defined only if the dot product of an outward pointing normal vector and a vector spanning from the data point to the facet is positive. The distance is measured from the point to the centroid of the facet

add_point_to_surface - Generate a tetrahedron from the data point and the facet. Add the tetrahedron to the surface, and remove the duplicated facet.

move_point_to_used_set - Label the point as used.

optimize_surface - Search through all the adjacent facet combinations. If the distance between the two uncommon points is less than the distance between the common points. Change the facets such that the uncommon points become the common points and vice versa. There is one exception to the rule. If the two uncommon points are primary points, and do not change the facets regardless of distance. The surface is considered fully optimized when the search goes through an entire cycle without changing any facets. (There is certainly opportunity for optimization of this part of the algorithm.)

Color Space Data With Corresponding 3-Color Input Device Signals

This is the most useful data set out of the three. To construct a full gamut descriptor, three things must be done: find the gamut surface, develop a function that will transfer us from input device space to color space, and an inverse function that will transform from color space back to device space. From this point on this function will be represented to as: $\Gamma\ (q_1,q_2,q_3)$, and the inverse function will be referred to as $\Gamma^{-1}(s_1,s_2,s_3)$

There are many functions that may be fitted to the data. A Bezier solid was chosen for two reasons: control points affect the entire solid and this helps to smooth out measurement noise, and there is an abundance of material written on the subject of Bezier curves.

Bezier solids are defined by control points, and since $\Gamma(q_1,q_2,q_3)$ has been defined to be a Bezier solid, the set of control points that best fits the data must be found. A least squares fit (O'Neil, 1987) will minimize the standard deviation of the error between $\Gamma(q_1,q_2,q_3)$ and the data. While this is a relatively simple task to perform, it is computationally demanding, requiring

the solution of an NxN matrix, where N is the number of control points in the solid.

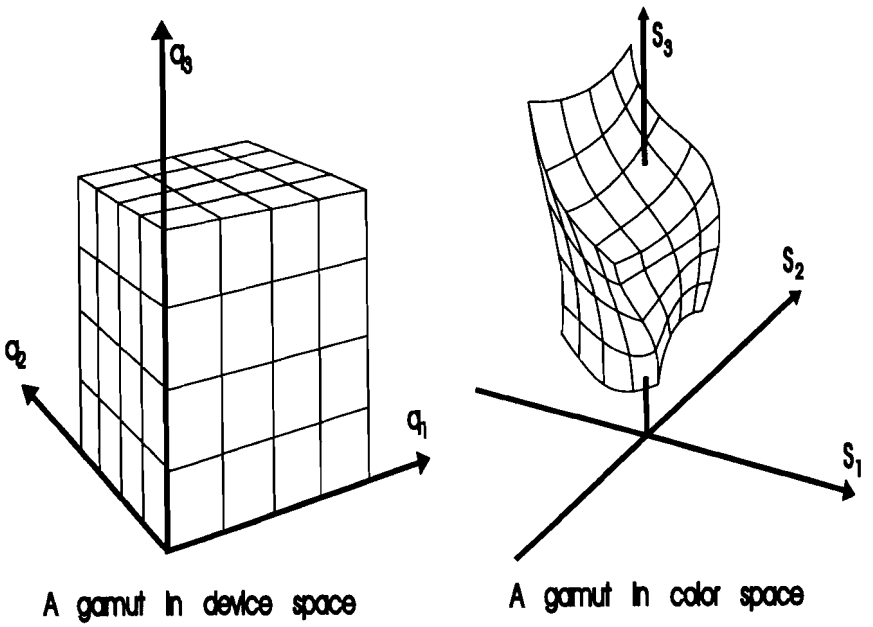

Figure 8. An example of how a gamut might map from device space to color space.

Effects of over-sampling and under-sampling

Over-sampling occurs when there are more data points than there are control points. This will most likely result in a $\Gamma(q_1,q_2,q_3)$ that does not exactly fit the data points. This is desirable because it reduces noise introduced by measurement errors. Under-sampling occurs when there are less data points than there are control points. Under-sampling is a problem because there is no longer a unique solution, and a singular matrix will result during the least squares fit. To remedy this, the Bezier solid was given an extremely small stiffness that will guarantee a unique solution for any reasonable size data sets.

A gamut surface can be approximated with the use of the function $\Gamma(q_1,q_2,q_3)$. The easiest way to accomplish this is to tile the gamut surface in device space with a large series of triangular facets. The three vertices for each facet can then be transformed into color space using the transfer function resulting in a tiled gamut surface in color space.

If $\Gamma(q_1,q_2,q_3)$ is well behaved, and there are no local minimums or maximums, a modified Newtonian convergence algorithm can be used as $\Gamma^{-1}(s_1,s_2,s_3)$. The first modification is to limit the number of iterations. If the data point is out of gamut there is no solution, and there will be no convergence. The second modification involves checking to see if the prediction is out of gamut. If the prediction is out of gamut, scale the correction by a factor of one half.

If the data is rectilinear (sampled at even intervals), a simpler approach can be used. A set of local cubic splines can be fitted to the data, instead of using a global fit of an arbitrary function. This will significantly reduce the number computations needed for fitting the curve to the data.

N-color gamut descriptor

This next section will discuss 4-color gamut descriptors, and the methodology can be generalized to n-color gamut descriptors. The technique is the same as described above, except for a few modifications that are required because there is no longer a one-to-one mapping of color space to device space.

First, a Bezier hypersolid is fit to the data points using a least squares method. This is a straightforward process, but computationally expensive, and generates a transfer function $\Gamma(q_1,q_2,q_3,q_4)$. It will map from device space (4-space) to color space (3-space).

To find the gamut surface, the gamut surface is first tiled in device space. Mathematically, this is a 4-dimensional cube, but most importantly, it is a simple closed surface. The device space gamut tiles are then mapped into color space. Since we have transformed from 4-dimensional space to 3-dimensional space our gamut in color space will no longer be a simple closed surface. Our tiles may now intersect each other. For exterior views of the gamut, any interior tiles will be hidden by the exterior tiles so that no more additional work is required. For internal or cut-away views of the gamut, the internal tiles need to be removed. By removing the enclosed tiles, the complex surface is reduced to a simple surface.

Since $\Gamma^{-1}(s1,s_2,s_3)$ may now be multi-valued, $\Gamma^{-1}(s1,s_2,s_3)$ becomes very difficult to model with an algorithm that converges to a single value. However, because a multi-valued result is undesirable, it is necessary to impose one or more constraints so that $\Gamma^{-1}(s_1,s_2,s_3)$ maps one to one. (For a CMYK system, this translates into defining K as a function C, M and Y). Then the modified Newtonian convergence algorithm is used as described above.

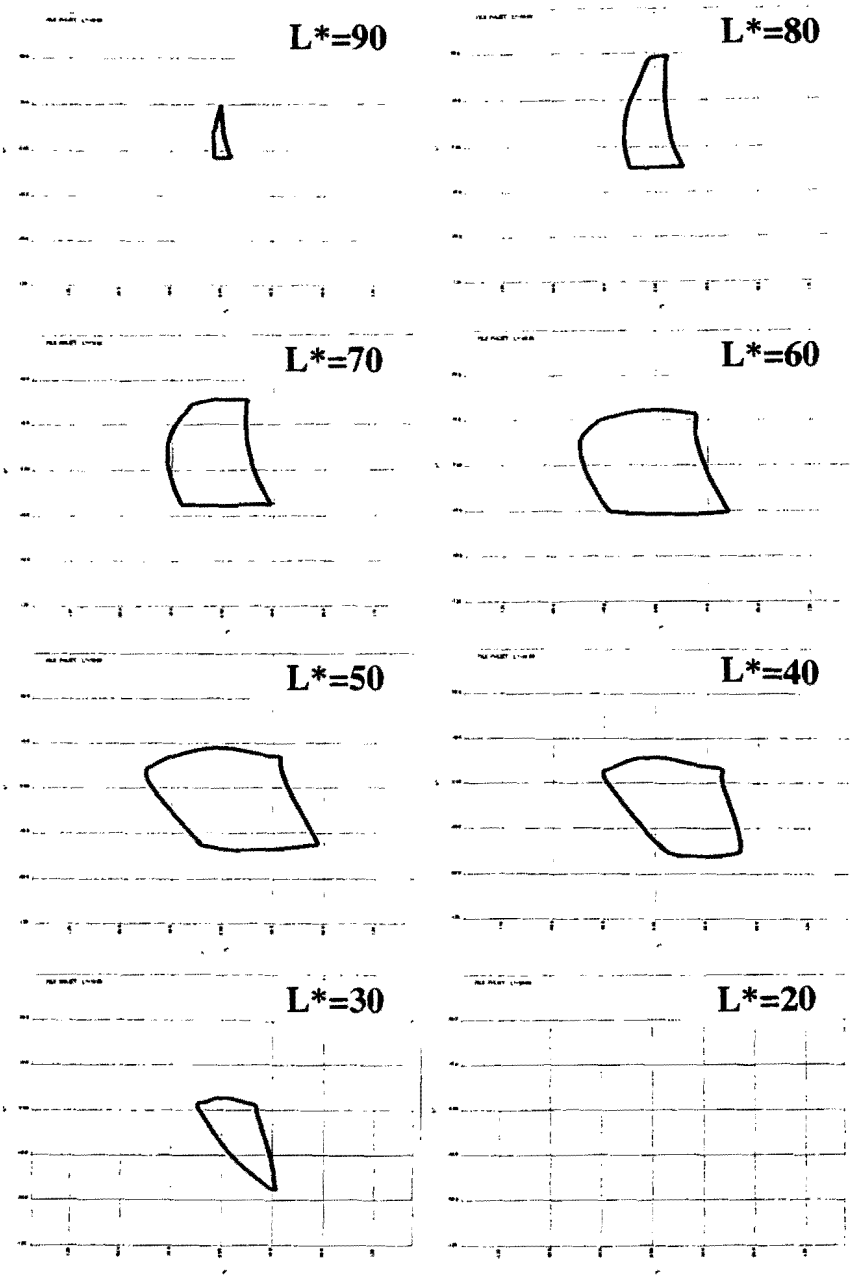

Figure 9. An ink jet gamut in CIELAB color space cut along planes of constant L*.

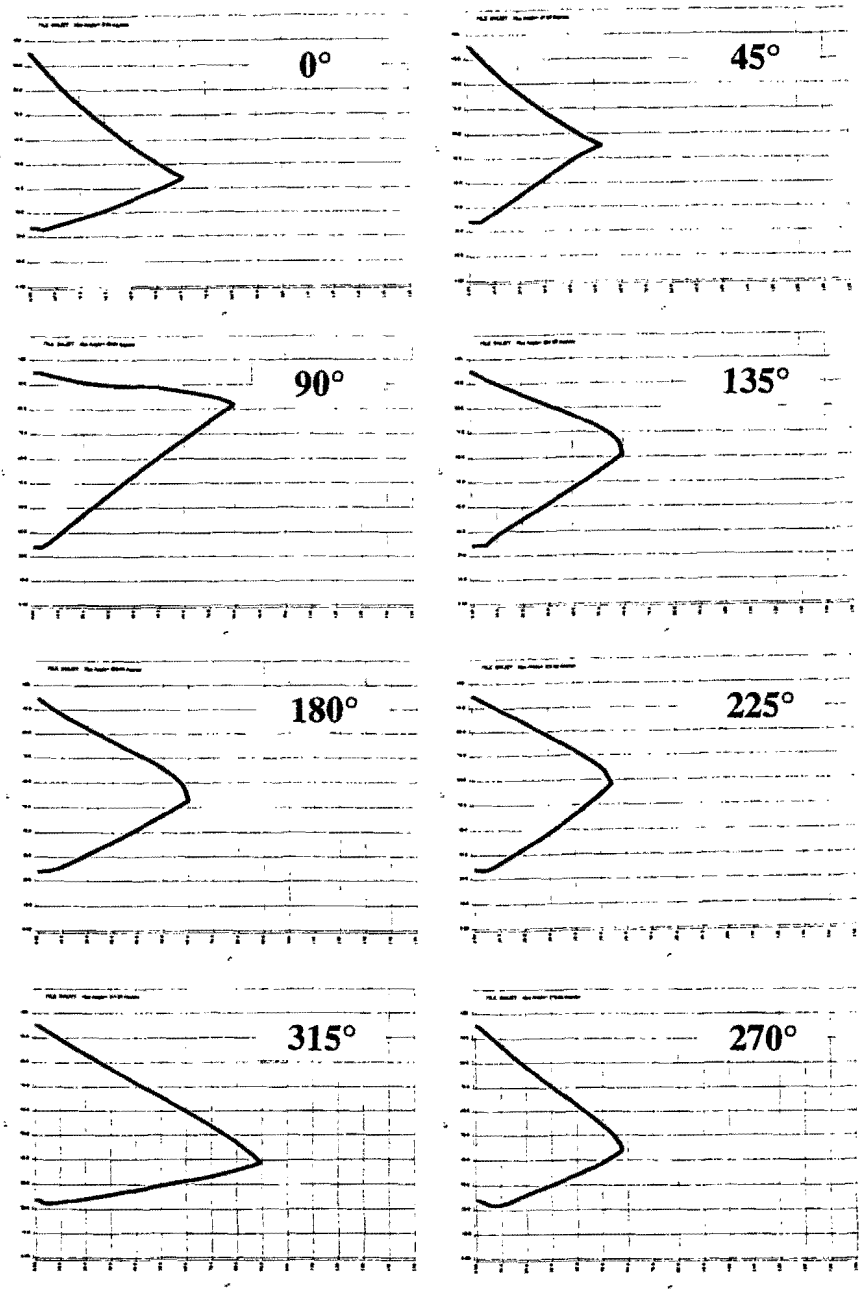

Figure 10. An ink jet gamut in CIELAB color space cut along various hue angle planes.

Summary

A technique for locating the color gamut of 3-color printing has been presented. There were suggestions about the extension of this technique for n-color printing The program, GAMUT, to determine an outermost surface when given an array of data points that are on the gamut surface. The program has been made to output a variety of formats for Wolfram's Mathematica, MathWorks' Matlab and general gamut descriptors for color management systems.

A commercially available graphical analysis tool, Matlab, has been used to plot the surface in colors that intuitively help the user in visualizing the device color gamut.

References

Engeldrum, P. G., Computing Color Gamuts of Ink Jet Printing Systems, *Procedures of the DID, Vol. 276/1*, 1986.

Meyer, John, Barth, Brian, Color Gamut Matching for Hard Copy, *SID 89 Digest*.

Bell, Ian E. and Cowan, William, Characterizing Printer Gamuts Using Tetrahedral Interpolation, *IS&T and SID's Color Imaging Conference: Transforms & Transportability of Color (1993)*.

Berns, Roy S., Spectral modeling of a Dye Diffusion Thermal Transfer Printer, RIT Munsell Color Science Laboratory, (1993).

Meyer, Gary W., Peting, Linda S. and Rakoczi, Ferenc, A Color Gamut Visualization Tool, *IS&T and SID's Color Imaging Conference: Transforms & Transportability of Color (1993)*.

O'Neil, Peter V., Advanced Engineering Mathematics, Second Edition, p1105, Wadsworth Publishing, 1987.

Appendix

Using The MathWorks Matlab for Gamut Visualization

The MathWorks Matlab program has been used to draw all the 3-dimensional figures shown in this paper. Matlab is easy to use and simple programs have been used to format files to be input directly to Matlab. The following example illustrates the Matlab file used to print Figure 5.

First gamut data is imported. Data is loaded from two sets of ASCII files with fixed length lines terminated with carriage returns and spaces separating the numbers. The first file contains the a* b* l* gamut data from an ink jet print engine and second the gamut data from a thermal wax engine.

The input data is of the form of three ASCII data files, created with extensions a, b, l. Each line of this file specifies the a* b* and L* coordinates of 2-dimensional polygons for each of n L* levels at hue increments m.

Each file will contain m+1 columns and n rows. For example, if the hue increments are 10 degrees, there will be 36 different hue positions and, to assure proper tiling of the 3-dimensional solid, the end point will be replicated.

```
load d:\matlab\gamuts\ink_jet.a
  a = ink_jet(:,:);
load d:\matlab\gamuts\ink_jet.b
  b = ink_jet(:,:);
load d:\matlab\gamuts\ink_jet.l
  l = ink_jet(:,:);

load d:\matlab\gamuts\t2_wax.a
  aa = t2_wax(:,:);
load d:\matlab\gamuts\t2_wax.b
  bb = t2_wax(:,:);
load d:\matlab\gamuts\t2_wax.l
  ll = t2_wax(:,:);
```

Surfaces can be cropped or made invisible by assigning particular matrix elements in the color array to NaN (Not a Number). First the indices of all matrix elements are found which are between arctan of 0 and 0.10. The L* elements at these indices are assigned NaN. This increases the visibility of the two gamuts and makes comparisons easier.

```
index = find (atan2(b,a) >  0.0 & atan2(b,a) < 0.10);
  l(index) = NaN * index;

index = find (atan2(b,a) > 0.25 & atan2(b,a) < 0.90);
  l(index) = NaN * index;
```

and so forth for the first 180 degrees of the a* b* diagram.

The following defines the gamuts from 180 to 360 degrees

```
index = find (atan2(b,a) <   0.0 & atan2(b,a) > -0.70);
```

```
    l(index) = NaN * index;
index = find (atan2(b,a) < -0.85 & atan2(b,a) > -1.50);
    l(index) = NaN * index;
```

The colormap matrix, x, can be constructed to give hues approximating CIELAB. The thermal wax gamut is shown as an entire surface. The ink jet gamut has been cut into angular strips and the two gamuts are further differentiated by different edgecolor specifications.

The colormap has 72 rows of which only 8 are shown here. The colors progress from magenta to red to yellow and so forth back to magenta.

```
c = - atan2(b, -a);
x = [1.00   0.0   0.90;
     1.00   0.0   0.80;
..... this data continues for blends from magenta to red
     1.00   0.0   0.05;
     1.00   0.0   0.00;
     1.00   0.05  0.00;
..... this data continues for blends from red to yellow
     1.00   1.0   0.0;
..... this data continues until magenta
     1.00   0.0   1.00;
     1.00   0.0   0.95];

clf
figure(1)
hold on
```

Hold the figure so that two gamuts are drawn on one plot.

```
colormap (x)
h=surf(a,b,l,c)
```

Set the colors of the edges of the ink jet gamut mesh to be a light gray.

```
set(h, 'edgecolor', [.7 .8 .7])
xlabel('a_star'), x1=get(gca, 'xlabel'), set(x1, 'FontSize',[10])
ylabel('b_star'), y1=get(gca, 'ylabel'), set(y1, 'FontSize',[10])
zlabel('l_star'), z1=get(gca, 'zlabel'), set(z1, 'FontSize',[10])

j=surf(aa,bb,ll,c)
```

Turn the grid on, set the bounds of the a* b* and L* axes and specify the viewpoint.

```
grid on, axis([-80, 80, -80, 120, 0, 100])
view(300, 70)
```

Set the colors of the edges of the thermal wax gamut to be dark gray.

```
set(j, 'edgecolor', [.25 .25 .25])
```

```
xlabel('a_star'), x1=get(gca, 'xlabel'), set(x1, 'FontSize',[10])
ylabel('b_star'), y1=get(gca, 'ylabel'), set(y1, 'FontSize',[10])
zlabel('l_star'), z1=get(gca, 'zlabel'), set(z1, 'FontSize',[10])
hold off
```

Draw four lines of titles.

```
axes ('vis', 'off', 'units','normal','pos', [0 0 1.0 1.04])
title('COLOR GAMUTS')
axes ('vis', 'off', 'units','normal','pos', [0 0 1.0 1.01])
title('Gamuts of ink jet versus thermal wax')
axes ('vis', 'off', 'units','normal','pos', [0 0 1.0 .98])
title('Gretag SPM50, D50, absolute white reference, white backing')
axes ('vis', 'off', 'units','normal','pos', [0 0 1.0 .95])
title('viewpoint: azimuth=300, elevation=70 (looking at cyan side)')

figure(2)
```

Another figure can be drawn with another viewpoint.