

A fast iteration algorithm for mapping L*a*b* to CMY values

Wilkin Chau and William B. Cowan
Department of Computer Science
University of Waterloo
Ontario, Canada

Abstract

Transforming CIE L*a*b* values to printer CMY values must be done when a colorimetric match is wanted for the printer output. The cost of such conversion is high. This paper describes a new efficient algorithm to perform this transformation. The computational cost is shown to be logarithmic in the number of sampling cells.

1. Introduction

Characterization of hard-copy output devices, such as digital colour printers, is normally performed by creating a set of samples with well-defined spacing in device coordinates (e.g., CMY) and measuring device-independent colour coordinates (e.g., L*a*b*) that correspond to each of the samples. The resulting measurements are used as a tabular transform: to get the L*a*b* value corresponding to a specific CMY value, first find the polyhedron, or cell, in which the desired CMY coordinates are located, then interpolate among the measured L*a*b* values at the vertices of the cell. Because the sampling in CMY is regular this process is efficient, with processing time to find the cell better than the logarithm of the number of cells, and a constant time algorithm for interpolation.

Very frequently the inverse transformation (finding device coordinates (CMY) corresponding to given device-independent coordinates (L*a*b*)) is implemented similarly. The cell-finding part of the procedure is, however, inefficient, with processing time varying linearly with the number of cells. This problem is well known; a variety of methods exist to improve its efficiency.

1. Resample to produce uniformly spaced cells in L*a*b* space. This process adds resampling error to measurement and interpolation error and its precision is often unacceptable.

2. Increase the measurement density. Measurement is, however, the most costly and error-prone part of the process. Furthermore, without resampling, increased measurement density means more cells, slowing the cell-finding algorithm.

3. Use data-fitting techniques to reduce resampling and measurement error.[1] This process seems to be the most promising alternative but the performance highly depends on the fitting model being used.

None of these solutions is fully effective. This paper provides a new cell-finding algorithm based on divide-and-conquer concept of binary subdivision. It finds the correct cell using an amount of processing that is logarithmic in the number of sampling cells. This performance does not depend on evenly spaced samples: the samples can be systematically displaced from uniform spacing. With this performance, the size of colour LUT can be increased. As a result, a higher accuracy conversion can be obtained by this approach. In this paper, we discuss how the algorithm works in detail. An analysis of its performance is also presented.

2. Basic Idea

The purposed algorithm provides a fast method to determine the CMY value corresponding to a given L*a*b* value within the printer gamut. It depends only on the forward transformation from a CMY value to its L*a*b* value being easily evaluated. In practice this is accomplished using a colour LUT followed by interpolation[2,3,4,5] or using a printer model.[6]

The basic idea of the proposed iteration algorithm is similar to the divide-and-conquer approach of binary subdivision. Initially, the algorithm uses the minimum and maximum colour values as the lower and upper bound for the colour components of the target CMY value. It starts with an initial CMY value, $(CMY)_0$, for which L*a*b* has been measured to have the value $(Lab)_0$. In the neighbourhood of $(Lab)_0$ the entire L*a*b* space is partitioned into regions, with each region having a well-determined relationship on one of the CMY coordinates. For example, one region has $C > C_0$, another has $M > M_0$, and so on (Figure 1). The colour space partition is well-approximated using only local information so it can be performed in constant time. Depending on which partition

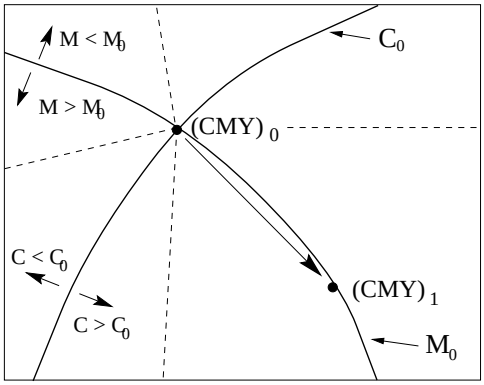

Figure 1: Regions defined by the contour curves of C_0 and M_0

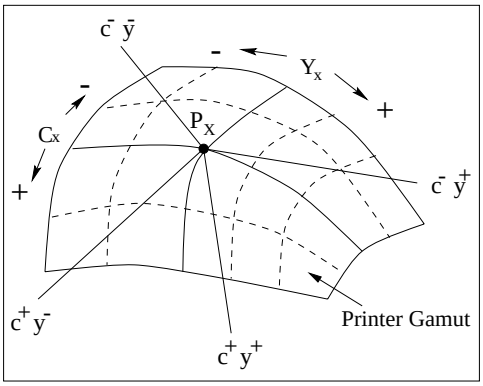

Figure 2: Partition of 2D printer gamut

contains the desired L*a*b* value, one of the colour components of $(CMY)_0$ becomes the lower or upper bound for the target CMY value. For instance, if the desired L*a*b* value has smaller C value then the C_0 is the upper bound for the C component. The mid-point of the current lower and upper bounds of each colour component is chosen to be the updated CMY value, $(CMY)_1$. This value has an improved estimate for whichever of the CMY values was indicated by the partition. This process continues until the point that has the closest CMY value related to the desired L*a*b* colour has been identified. Interpolation is then used to find the correct CMY value within the cell.

As described above, there are three basic operations in each iteration, namely, gamut partition, region detection, and point advancement. These operations are discussed in details on the next few sections.

3. Gamut Partition

Since fast convergence depends on the good region subdivision, a robust partition method is essential. The following partition method is based on geometric features of printer gamuts in CIE L*a*b* space.

Consider the set of CMY values having a specified value of a particular colour component, say C. These colour values define a plane in the CMY space, which corresponds to a smooth continuous surface (iso-surface) in CIE L*a*b* space. As the values of the M and Y components change across the surface, their L*a*b* values vary continuously. The curvatures of such surfaces are usually but not negligible. These assumption, which holds for well-behaved printers, underpins the partition method described below.

To simplify the description of the partition procedure, let us first consider the 2 dimensional case. Imagine that there is a 2D device independent colour space, similar to the CIE L*a*b* space, covering all the colours generated by the two inks, for example cyan (C) and yellow (Y). Any printer using only these two inks has a gamut that is a closed region in the colour space. Figure 2 shows a possible printer gamut. Each curve in the figure is a contour curve for a constant C or Y value. Now consider a point P_x inside the printer gamut, with C_x and Y_x its C and Y component. The printer gamut can be divided into four regions based on the location of P_x. Each region contains points that have a fixed relationship with one of the colour components of P_x. One such partition is shown in Figure 2, in which the regions are defined by four boundary lines c^-y^-, c^-y^+, c^+y^-, and c^+y^+. The points inside the region bounded by the lines c^-y^- and c^-y^+ have C smaller than C_x, while the points inside the region bounded by the c^+y^+ and c^-y^+ have Y larger than Y_x, and so on. In general, each region corresponds to a portion of the gamut that has either smaller or larger amount of one of the colour components than that of P_x . A partition is not unique, but every partition must meet the following two conditions: (1) the boundary lines do not cross the contours defined by $C = C_x$ and $Y = Y_x$, (2) only two boundary lines lie on the same side of the region separated by the C_x or Y_x contour curves. The first condition ensures that all points from the same side of a boundary line have the same relationship with one of the colour components of P_x, and the second condition ensures that each one of four gamut regions bounded by the two contour curves has one and only one boundary line in it. As a result, the segment of contour line that runs from P_x to one of the gamut border is inside the region bounded by two neighouring boundary lines.

Applying the above criteria strictly, it is very costly to define a suitable set of boundary lines. A better alternative defines the boundary lines based on local values of P_x. Because printers ues inks with chromaticities that are far apart in the colour space, the contour curves of different inks are almost perpendicular to each other in the region close to P_x. If the lines that evenly divide the cross contour angles are chosen to be the boundary lines, the criteria are met locally. The local region is very large if the curvature of the contours is small, as it normally is. (Figure 3)

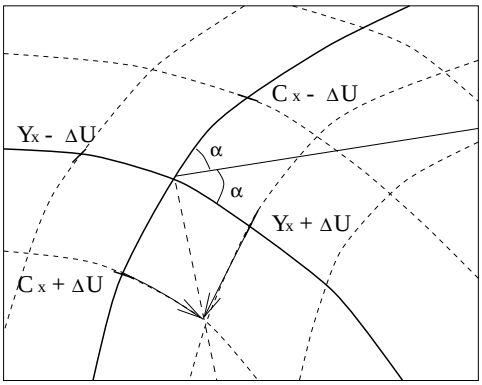

Figure 3: Boundary lines of partition the of 2D printer gamut

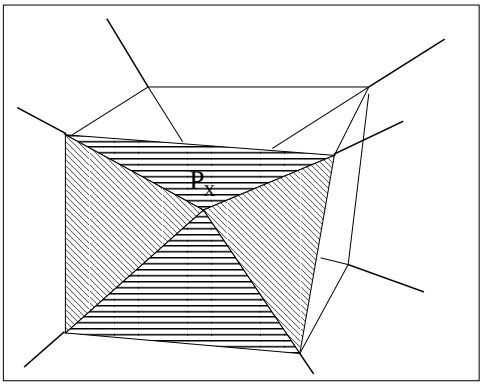

*Figure 4: Boundary planes of gamut partition in L*a*b* space*

To obtain the angles of two cross contour curves, the tangents of the contours at the cross point have to be computed. When the mathematical model of a given printer is not available, it is impossible to compute the tangents in general. However, the above boundary lines are by no means the only solution for partitioning the gamut; any close approximations of the above boundary lines serves equally well. One workable solution uses the four lines that connect the current P_x point to the point $(C_x - \Delta U, Y_x - \Delta U)$, $(C_x - \Delta U, Y_x + \Delta U)$, $(C_x + \Delta U, Y_x - \Delta U)$ and $(C_x + \Delta U, Y_x - \Delta U)$, respectively, where ΔU is a fixed value.

Partitioning the printer gamut in the 3D CIE L*a*b* space is analogous to the 2D case described above. Instead of four boundary lines, twelve boundary planes are using to partition the gamut. Each region in the gamut is confined by four planes (Figure 4). The criteria for choosing those planes are defined as follows: (1) the boundary planes do not cross any iso-surfaces of the colour components of P_x, and (2) four and only four planes lie on the same side of the region separated by each iso-surfaces. Similar to the 2D case, one possible way to partition the gamut is to use the boundary planes defined by two neighbouring lines which connecting the current point P_x to any two points in the set $\{(C_x \pm \Delta U, M_x \pm \Delta U, Y_x \pm \Delta U)\}$ that differ by only one component. For example, one boundary plane is defined by the line connecting P_x to $(C_x - \Delta U, M_x - \Delta U, Y_x - \Delta U)$ and another connecting P_x to $(C_x - \Delta U, M_x - \Delta U, Y_x + \Delta U)$. With the set of eight points $\{(C_x \pm \Delta U, M_x \pm \Delta U, Y_x \pm \Delta U)\}$, twelve planes are defined to partition the gamut in the CIE L*a*b* space. If P_x lies on the boundary of gamut, the outside gamut boundary planes can be obtained by extrapolating the inside gamut planes through P_x.

The value chosen for ΔU is not critical. In practice, we choose for ΔU the interval used to build the colour LUT. The boundary planes defined based on this ΔU value accurately partition the gamut for any colour point P_x.

4. Region Detection

Upon defining the regions, the next task is to find out which region the target point P_t is in. Based on this information, the direction for point advancement can be determined. Since each region is bounded by four planes and each plane divides the colour space into two half spaces, the region detection problem can be viewed as identifying which half space the point is in for each boundary plane. By using a fixed normal vector convention, and computing the dot product of the normal vector of a plane and the vector, defined by the difference of the point P_x and P_t, the half space in which the target point locates can be identified. For example, if the sign of the dot product result is positive, then P_t lies on the same side of the normal vector; otherwise, P_t is on the opposite side. Repeating this calculation for each boundary plane, the partition region that P_t is in is found.

5. Point Advancement

After knowing which region P_t is in, the components of P_x can be updated. If, for instance, the P_t is in the region in which all points have less magenta component than the P_x, then the magenta component of P_x should be reduced to define the new P_x. Note that, in this case, the current magenta component can be used to define the upper bound of the magenta component for P_t. Conversely, if the magenta component is too small, it should be increased to define the new P_x. The current level of the magenta component is the lower bound for P_t. The amount of change to the colour component is calculated based on the current upper and lower bound values of the colour component. The new colour component value is most effectively placed at the mid-point of the bounding values. When the colour relationship between the CMY and L*a*b* spaces is defined by the colour LUT, the L*a*b* value of the mid-point

Table 1: Pseudocode of the iteration algorithm

1. Set up the initial point of P_x and compute its Lab values
2. If P_x = P_t, go to step (11)
3. Determine 12 boundary planes based on the 8 neighbour points of P_x
4. Compute the normal vector of each plane
5. Compute the dot products of the vector (P_t - P_x) with the normal vector of each plane
6. Determine the region in which P_t lies
7. If P_t lies in the outside gamut region, go to step (12)
8. Adjust the lower or upper bound of P_t based on the result of (6)
9. Update the colour component of P_x based on the bounding values of (8)
10. Go to step (2)
11. Return the CMY value of P_x
12. Return out of gamut status

may not be available without interpolation. In this case, the nearest sample point is used as the new P_x instead. Notice that the position of the new P_x is necessarily closer to the target point compared to the current P_x. The operations of gamut partition, region detection and point advancement are repeated until the point that has the closest CMY value related to the desired L*a*b* colour has been identified. Interpolation is then used to find the correct CMY value within the cell.

6. Performance Analysis of the Algorithm

Table 1 is the pseudocode of the proposed algorithm. Steps 2 to 10 are performed for each iteration. The total number of iterations depends on the precision required. As mentioned in the previous section, the new colour component value is the mid-point of its upper and lower bound. This approach is similar to the binary search operation which has order of $\log(M)$, where M is the number of points being searched. To determine the final CMY value, this algorithm requires $O(\log(N))$ iterations, where N is the number of sample points for each colour component. For printer with 256^3 different colour values (eight bits per primary) the maximum number of iterations needed to find the CMY value is $3\log(256) = 24$.

A way to further improve the performance is to try to reduce the number of iterations. Since the operation is normally performed on pixel by pixel basis, and the neighbouring pixels in an image are often very similar to each other. The number of iterations can be reduced in most cases if the CMY value of the last processed pixel is used as the initial value of P_x.

7. Conclusion

A fast iteration algorithm that maps L*a*b* to CMY value was described. Its computation cost is logarithmic in the number of the sampling cells, and its performance is robust provided that the underlying CMY to L*a*b* mapping is sufficiently smooth. We have shown experimentally that printer models currently in use are adequately smooth. Measurement error can upset this condition, of course, and the gamut fitting techniques[4] previously investigated in our laboratory can be used to condition measured data so that our new algorithm is guaranteed to converge. Throughout the discussion in this paper we assume a colour LUT is used for colour space conversion. In fact, this algorithm works equally well when a mathematical printer model is used instead.

This algorithm has been described in terms of CMY and L*a*b*, common colour spaces used in digital colour printing. It works equally well for other spaces, and for any inversion of a tabular function that is sufficiently well- behaved. Investigation of the precise conditions under which it onverges robustly is continuing.

The algorithm is also attractive for evaluation of table-defined functions where the underlying sampling is not regular. Thus, it is likely to be useful for evaluation of transforms of dark colours, where adaptive sampling is particularly beneficial.

8. References

1. H. R. Kang, *Color Technology for Electronic Imaging Devices*, SPIE Optical Engineering Press, 1997.
2. H. R. Kang, "Comparisons of three-dimesional interpolation techniques by simulations," Proc. SPIE **2414**, 104-114 (1995)
3. P. Hung, "Colorimetric calibration in electronic imaging devices using a look-up-table model and interpolations," *Journal of Electronic Imaging*, **2(1)**, 53-61 (1993)
4. I. E. Bell and W. Cowan, "Device characterization using spline smoothing and sequential linear interpolation," *Proc. IS&T and SID's 2nd Color Imaging Conference: Color Science, Systems and Application*, Nov. 1994, 29-33.
5. J. M. Kasson, W. Plouffe, and S. I. Nin, "A tetrahedral interpolation technique for color space conversion," Proc. SPIE **1909**, 127-138 (1993)
6. R. S. Berns, "Spectral modeling of a dye diffusion thermal transfer printe r," *Journal of Electronic Imaging*, **2**: p.359-370 (1993).