Behavior Research Methods, Instruments, & Computers
1989, 21 (5), 502-524

Lookup tables for linear trajectories through color space

HAROLD STANISLAW
University of New South Wales, Kensington, New South Wales, Australia

Many applications require changing the color of a visual stimulus, so that a linear trajectory through color space is traversed. The present paper outlines two methods for generating such trajectories on computer-controlled video systems. FORTRAN implementations of each method are provided, along with measures of interest for studies of color vision.

Ramping a colored stimulus on or off requires holding chromaticity constant while luminance is changed. Other applications hold luminance constant while chromaticity is changed (e.g., Olzak, Thomas, & Stanislaw, 1987). Both of these manipulations require color changes that describe a linear trajectory through color space. They can be generated in real-time on a computer-controlled color video system, by filling the system's *color lookup table* (LUT) with the desired trajectory, and then successively changing the LUT entry that is used to "paint" the stimulus.

In general, the change from one color to the next should be minimal. This is especially important when thresholds (e.g., for detecting chromatic shifts) are being measured, because the precision of such measurements is limited by the magnitude of the color change.

Halftoning can reduce the difference between successive colors to any level desired (Mulligan, 1986). However, halftoning cannot be used with small stimuli, and researchers may wish to avoid the complications it introduces. The present paper details methods for filling an LUT with entries that describe a given trajectory through color space and differ from each other as little as possible without halftoning.

Terminology

The discussion below applies generally to any color-measurement system, but it is based on the 1931 Commission Internationale de l'Éclairage (CIE) *XYZ* system. This system describes colors in terms of their red-green (X), achromatic (Y), and yellow-blue (Z) components, collectively labelled *tristimulus coordinates* (Pokorny & Smith, 1986). LUT entries are also trivariate, describing the voltages that drive red (R), green (G), and blue (B) electron guns.

I thank two anonymous reviewers for providing helpful comments on an earlier version of this manuscript. Correspondence regarding this paper should be addressed to Harold Stanislaw, School of Psychology, University of New South Wales, P.O. Box 1, Kensington, NSW 2033, Australia.

The *XYZ* and *RGB* color spaces are not isomorphic, but *RGB* entries can be converted to tristimulus coordinates, using the equation

$$[X\ Y\ Z] = [e_R\ e_G\ e_B]\begin{bmatrix} x_r & y_r & z_r \\ x_g & y_g & z_g \\ x_b & y_b & z_b \end{bmatrix}, \quad (1)$$

where the rightmost matrix describes the chromaticity coordinates of the monitor's phosphors (obtainable from the manufacturer), and the es are the phosphor excitations that result from loading the LUT with entries $[RGB]$. *Gamma correction* must be used to determine the excitations that result from a given set of LUT entries. Gamma functions are nonlinear, and vary across time and monitors. However, monitors can be calibrated with a photometer, and the calibration data can be used to construct arrays containing values for e_R, e_G, and e_B, indexed by the LUT entry (for details, see Brainard, 1989; Cowan, 1983; Mulligan, 1986; Watson et al., 1986). Alternately, calibration data may be used to derive a formula for gamma correction. The formula used here is

$$e = \begin{cases} \exp[p_0+p_1(\ln v)+p_2(\ln v)^2+p_3(\ln v)^3] & \text{if } v > v_0 \\ q_0+q_1v+q_2v^2 & \text{if } v \le v_0 \end{cases} \quad (2)$$

where e is the excitation, v is the LUT entry, p_0-p_3 are parameters for a power function, q_0-q_2 are parameters for a quadratic function, and v_0 is the voltage setting marking a transition from a power to a quadratic function. Parameter values for one video system are listed in Appendix A, but these values should not be generalized to other systems.

To convert tristimulus coordinates to LUT entries, both sides of Equation 1 can be postmultiplied by the inverse of the phosphor chromaticity coordinates:

$$[e_R\ e_G\ e_B] = [X\ Y\ Z]\begin{bmatrix} x_r & y_r & z_r \\ x_g & y_g & z_g \\ x_b & y_b & z_b \end{bmatrix}^{-1}. \quad (3)$$

Copyright 1989 Psychonomic Society, Inc.

The corresponding LUT entries can then be determined through *inverse gamma correction*. This may be accomplished with an array of voltage settings indexed by excitation values (constructed on the basis of calibration data), or a suitable prediction equation. Alternately, if an equation is used for gamma correction, the roots of that equation may be used for inverse gamma correction. For example, subroutine XYZRGB (listed in Appendix A), finds the roots of Equation 2, using the method detailed by Press, Flannery, Teukolsky, and Vetterling (1986, pp. 145-146).

Measures of Interest

Consider a visual stimulus that is initially at tristimulus coordinate (X_0, Y_0, Z_0), and then changes in color to follow a particular linear trajectory through color space. The trajectory may be described by a vector, **m**, of the form $[x, y, z]$. Each entry in **m** denotes the relative change from the initial color along a particular axis of color space. For example, $[1,0,0]$ describes a change in X only, $[1,0,1]$ describes equal changes in both X and Z, and $[X_1-X_0, Y_1-Y_0, Z_1-Z_0]$ describes a change from the initial color to tristimulus coordinate (X_1, Y_1, Z_1).

Filling an LUT with entries that satisfy **m** would be a trivial exercise, except that LUT entries must be integers. This limits the colors that can actually be displayed by a given video system (see Figure 1), so that **m** cannot be traversed precisely. Instead, the colors produced by each LUT entry may deviate from the desired trajectory, producing *quantization errors*.

This problem is illustrated in Figure 2. The initial color, O, is at (X_0, Y_0, Z_0). A desired color, D, falls on vector **m**, but it cannot be displayed by the apparatus. The nearest color that actually can be displayed, A, is at (X_1, Y_1, Z_1). However, quantization error is not necessarily the distance from A to D, because there may be another point that also lies on **m** but is closer to A. This point, P, is at (X', Y', Z'), where A projects onto **m**. The quantization error is thus $\overline{AP}$.

Lines $\overline{AP}$ and $\overline{OP}$ are perpendicular, so

$$\overline{AP} = \sqrt{\overline{OA}^2 - \overline{OP}^2}. \quad (4)$$

$\overline{OA}$ is simply

$$\overline{OA} = \sqrt{(X_1-X_0)^2 + (Y_1-Y_0)^2 + (Z_1-Z_0)^2}, \quad (5)$$

while calculus can be used to find the distance $\overline{OP}$ that minimizes $\overline{AP}$:

$$\overline{OP} = \frac{(X_1-X_0)x + (Y_1-Y_0)y + (Z_1-Z_0)z}{\sqrt{x^2+y^2+z^2}}. \quad (6)$$

$\overline{OA}$ and $\overline{OP}$ are both measures of interest. Suppose that O is the true (i.e., displayed) background color, and stimuli painted with color A are just detectable against this background. $\overline{OA}$ then represents the threshold for detecting colors that vary from background O along the vector leading from O to A. This vector is ideally identical to **m**. Note that the threshold is not $\overline{OP}$. However, thresholds can be measured most precisely if $\Delta\overline{OP}$—the change in

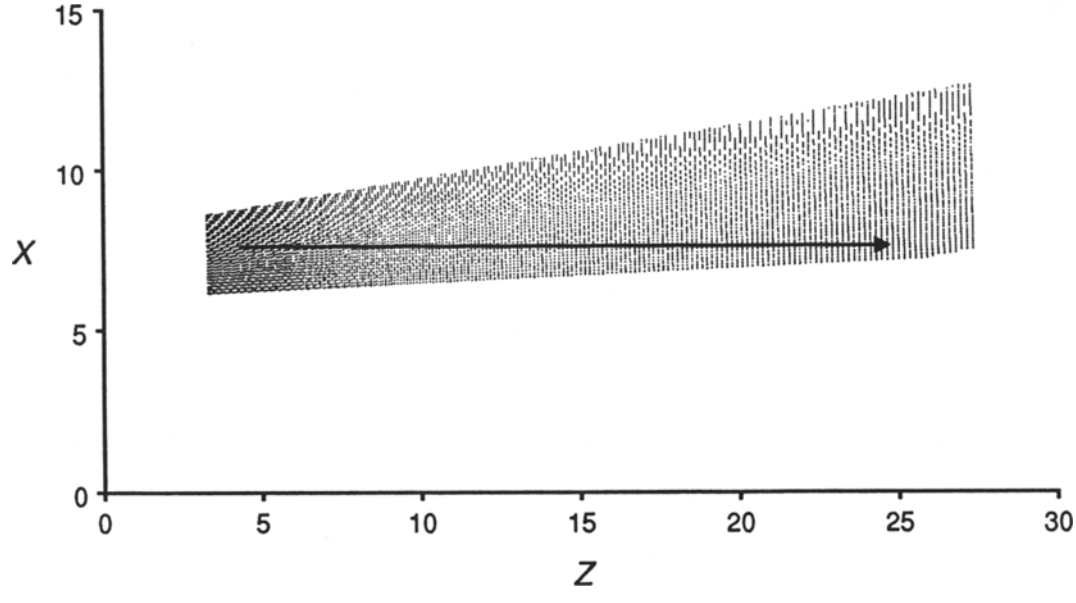

Figure 1. Section through the *XZ* plane (with $Y = 10.00 \pm 0.01$) of the color space for a Barco CDCT 5137 monitor driven by a CAT 1631 graphics board in a Cromemco CS-2 computer. This system can display 2^{24} colors, of which 9698 (0.06%) are shown. The arrow starts at (7.6, 10.0, 4.3) and follows the trajectory m = [0, 0, 1].

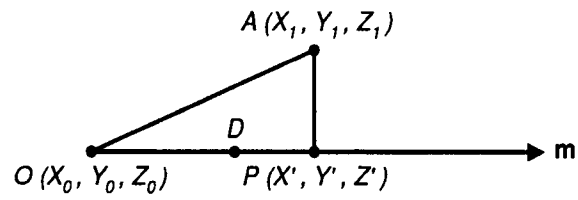

Figure 2. Measures of interest for a trajectory through color space from point *O* along vector m. See text for details.

$\overline{OP}$ from one LUT entry to the next—is as small as the apparatus permits. Note also that $\Delta\overline{OP}$ must be positive throughout the LUT for movement through color space to be unidirectional.

Filling an LUT with a Grid Search

One method for filling an LUT is to start with the set of displayable colors, and select the subset of desirable colors. This method can be implemented by starting with the LUT entry for the initial color, searching a surrounding grid of candidate entries, and using gamma correction to find the tristimulus coordinates (and thus $\overline{OP}$ and $\overline{AP}$) for each candidate. Candidates that do not increase $\overline{OP}$ can be rejected. The best remaining candidate can then be adopted as the next LUT entry, used as the starting point for another grid search, and so on, until the LUT is full or the limits of the apparatus are met.

The "best" candidate is that which produces the smallest positive $\Delta\overline{OP}$, but nevertheless yields a quantization error that is within some acceptable level, $\overline{AP}_{max}$. If no such candidate can be found, the grid may be expanded to search a greater portion of the displayable color space.

This method is implemented by FORTRAN subroutine GRID (see Appendix B). GRID takes **m**, the initial color, and $\overline{AP}_{max}$ as input. Inverse gamma correction is used to find the first LUT entry. Subsequent entries are found by using gamma correction to evaluate a grid that initially allows each value of *R*, *G*, and *B* to vary by ± 1 from the previous entry. The grid expands as needed to allow a maximum variation of ± 10.

GRID spends most of its time making gamma corrections. This involves converting LUT entries into excitations, and excitations into tristimulus coordinates. Since the former conversions involve far more calculations than the latter, GRID considers factorial combinations of the excitations only, not of the LUT entries themselves. This reduces execution time dramatically. Further reductions can be obtained by recalculating the contribution of the red and green excitations to the tristimulus coordinates only when the corresponding excitation changes, not when *any* excitation changes. (This modification is not implemented in Appendix B, because it reduces the listing's interpretability.)

Filling an LUT by Vector Addition

An alternate method for filling an LUT is to begin with a set of desired colors, and select the subset that can be displayed. The desired colors can be generated by adding **m** in an iterative manner to the initial color. Inverse gamma correction can then determine the LUT entries needed to display each of these colors.

Vectors containing elements that are proportional to each other—such as [1,2,3] and [2,4,6]—should produce identical LUTs. This can be ensured by rescaling **m** to an arbitrary standard. In general, **m** should be minimized, so that $\Delta\overline{OP}$ is minimized. However, there is a limit beyond which further reductions in **m** merely increase roundoff error and the number of iterations needed to produce novel LUT entries. This limit may change throughout a given trajectory.

Another problem is that $\overline{OP}$ may decrease from one LUT entry to the next, even though each desired color is farther along **m** than the previous color. Thus, the LUT must occasionally be reordered, to ensure that $\Delta\overline{OP}$ is always positive.

Vector addition is used by FORTRAN subroutine ADDM (see Appendix C). ADDM first standardizes **m** so that the most extreme entry has an absolute value of unity. The LUT produced by ADDM is thus affected only by the relative proportionality of the entries in **m**, not the values themselves. ADDM subsequently rescales **m** as needed. If the LUT entry changes after only a single addition of **m**, **m** is deemed too large. ADDM then "backs up" a step and halves **m** before proceeding further. If the LUT entry does not change after two additions of **m**, **m** is deemed too small and therefore doubled. All rescaling of **m** is in powers of 2, to control roundoff error.

ADDM's inverse gamma function returns fractional LUT entries, which are converted to integers by adding 0.5 and rounding down. This method does not always find the displayable color that is nearest the desired color. A grid search can therefore follow inverse gamma correction, to determine whether rounding should be up or down. In practice, however, this does not improve the LUT.

Comparison of Methods

Typical quantization errors from GRID and ADDM are shown in Figure 3. The quasicyclic variation of the errors is a form of aliasing: The displayable colors are distributed in an approximate grid, and the trajectory cuts diagonally across this grid (see Figure 1).

Figure 3 also illustrates a form of heteroscedasticity: The quantization errors produced by GRID tend to increase with $\overline{OP}$. This occurs partly because GRID chooses the best candidate for the current iteration, which may be suboptimal for future iterations. GRID thus takes occasional "wrong turns." These are corrected by grid expansion when they become excessive, but heteroscedasticity is unavoidable for both routines when a trajectory

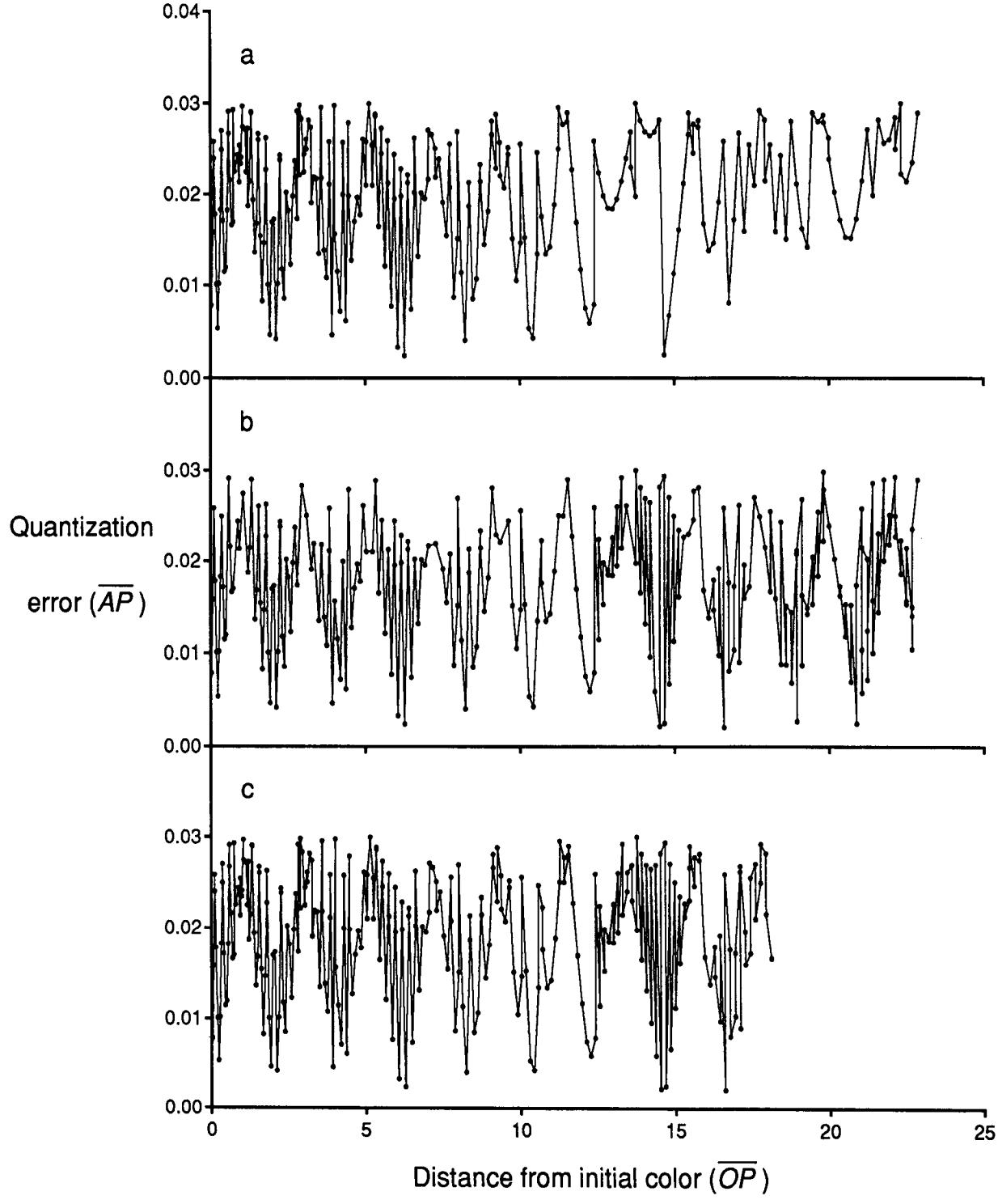

Figure 3. Quantization errors for the first 256 LUT entries produced by (a) GRID, (b) ADDM, and (c) the union of GRID and ADDM, for the trajectory illustrated by the arrow in Figure 1. $\overline{AP}_{max}$ was set to 0.03.

approaches the limits of the apparatus, where neighboring colors are farther apart (e.g., the upper right portion of Figure 1).

The average values of $\overline{AP}$ and $\Delta\overline{OP}$ are comparable for both GRID and ADDM, despite differences in the LUTs. GRID produces smaller color changes than ADDM at small values of $\overline{OP}$, while ADDM is superior at large values. However, experience with other trajectories and initial colors suggests that neither routine consistently outperforms the other.

Execution times for the two routines are also similar. Both routines fill the LUT for Figure 3 in 4 sec on a 16-MHz computer equipped with an optimizing compiler and math coprocessor. This drops to 3 sec if CALL state-

ments are eliminated by incorporating the gamma and inverse gamma functions directly within each routine. Clearly, however, neither routine can generate new LUTs in real time (e.g., in response to observer inputs), particularly if implemented on a slower computer.

Given these similarities, the choice between routines may be dictated by the availability and accuracy of methods for gamma and inverse gamma correction. Both routines involve both corrections, but GRID centers on the former, while ADDM centers on the latter. In fact, if the LUT entry for the initial color can be specified a priori, inverse gamma correction can be eliminated from GRID. Similarly, ADDM can ignore gamma correction if the specific values of $\overline{AP}$ and $\overline{OP}$ are not of interest (although in this case $\Delta\overline{OP}$ may occasionally be negative).

Note also that GRID is highly dependent upon the value of $\overline{AP}_{max}$, while ADDM is not. In fact, ADDM performs well even if $\overline{AP}_{max}$ is unspecified or absurdly large, while GRID does not. Thus, ADDM should be used if there is no interest in the actual magnitude of the quantization errors.

If execution time is not of concern, both routines can be used. The LUT for Figure 3c, for example, is the union of the LUTs produced by GRID and ADDM. It yields a smaller average $\Delta\overline{OP}$ than either GRID or ADDM (as indicated by the reduction in the final value of $\overline{OP}$), and combines the best elements of both approaches.

REFERENCES

BRAINARD, D. H. (1989). Calibration of a computer controlled color monitor. *COLOR Research & Application*, **14**, 23-34.

COWAN, W. B. (1983). An inexpensive scheme for calibration of a colour monitor in terms of CIE standard coordinates. *Computer Graphics*, **17**, 315-321.

MULLIGAN, J. B. (1986). Minimizing quantization errors in digitally-controlled CRT displays. *COLOR Research & Application*, **11**, S47-S51.

OLZAK, L. A., THOMAS, J. P., & STANISLAW, H. (1987). *Development of a chromatic/luminance contrast scale* (Contract No. DTCG-39-C-80205). Washington, DC: U.S. Coast Guard.

POKORNY, J., & SMITH, V. C. (1986). Colorimetry and color discrimination. In K. Boff, L. Kaufman, & J. P. Thomas (Eds.), *Handbook of perception and human performance: Vol. 1. Sensory processes and perception* (pp. 8-1-8-51). New York: Wiley.

PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., & VETTERLING, W. T. (1986). *Numerical recipes: The art of scientific computing*. Cambridge, England: Cambridge University Press.

WATSON, A. B., NIELSEN, K. R. K., POIRSON, A., FITZHUGH, A., BILSON, A., NGUYEN, K., & AHUMADA, A. J., JR. (1986). Use of a raster framebuffer in vision research. *Behavior Research Methods, Instruments, & Computers*, **18**, 587-594.

APPENDIX A
Subroutines to Convert between *RGB* and *XYZ* Color Space

```
      SUBROUTINE RGBXYZ(LUTR,LUTG,LUTB,X,Y,Z)
C
C FORTRAN subroutine to convert from RGB color space (the LUT entries
C LUTR, LUTG and LUTB--all INTEGER*2 variables) to XYZ color space
C (tristimulus coordinates X, Y and Z--all REAL*4 variables).  All
C internal calculations are in double precision.
C
C ** WARNING ** The parameter values in this subroutine should not be
C generalized to other video systems.
C
      REAL*8 V(3),E(3),V0(3),P0(3),P1(3),P2(3),P3(3),Q0(3),Q1(3),Q2(3)
      DATA V0/0.11764310D+03,0.14000000D+03,0.10949059D+03/
      DATA P0/0.15284195D+02,-.74795130D+02,0.39596094D+02/
      DATA P1/-.11514202D+02,0.39700333D+02,-.24230622D+02/
      DATA P2/0.25233193D+01,-.71598417D+01,0.48892751D+01/
      DATA P3/-.15663061D+00,0.45336301D+00,-.30539627D+00/
```

APPENDIX A (Continued)

```
      DATA Q0/0.62875237D-01,0.51273829D-01,0.20635986D+01/
      DATA Q1/0.17687158D-02,0.10497312D-02,0.10010797D-02/
      DATA Q2/0.13656599D-03,0.16917282D-03,0.43687862D-03/
      V(1)=LUTR
      V(2)=LUTG
      V(3)=LUTB
C
C Convert LUT entries to excitations, using a quadratic function for
C small entries, and a power function for large entries.
C
      DO 100 I=1,3
      IF(V(I).LE.V0(I)) THEN
        E(I)=Q0(I)+V(I)*(Q1(I)+V(I)*Q2(I))
      ELSE
        VLOG=LOG(V(I))
        E(I)=EXP(P0(I)+VLOG*(P1(I)+VLOG*(P2(I)+VLOG*P3(I))))
      ENDIF
100   CONTINUE
C
C Post-multiply the excitations by the monitor chromaticity coordinates,
C to convert the excitations to tristimulus coordinates.
C
      X=0.62D0*E(1)+0.210D0*E(2)+0.15D0*E(3)
      Y=0.33D0*E(1)+0.675D0*E(2)+0.06D0*E(3)
      Z=0.05D0*E(1)+0.115D0*E(2)+0.79D0*E(3)
      RETURN
      END

      SUBROUTINE XYZRGB(X,Y,Z,IERR,LUTR,LUTG,LUTB)
C
```

APPENDIX A (Continued)

```
C FORTRAN subroutine to convert from XYZ color space (the tristimulus
C coordinates X, Y and Z--all REAL*4 variables) to RGB color space (the
C LUT entries LUTR, LUTG and LUTB--all INTEGER*2 variables).  All internal
C calculations are in double precision.
C
C ** WARNING ** The parameter values in this subroutine should not be
C generalized to other video systems.
C
      IMPLICIT REAL*8 (A,B,E,T)
      REAL*8 V(3),E(3),V0(3),P0(3),P1(3),P2(3),P3(3),Q0(3),Q1(3),Q2(3)
      DATA V0/0.11764310D+03,0.14000000D+03,0.10949059D+03/
      DATA P0/0.15284195D+02,-.74795130D+02,0.39596094D+02/
      DATA P1/-.11514202D+02,0.39700333D+02,-.24230622D+02/
      DATA P2/0.25233193D+01,-.71598417D+01,0.48892751D+01/
      DATA P3/-.15663061D+00,0.45336301D+00,-.30539627D+00/
      DATA Q0/0.62875237D-01,0.51273829D-01,0.20635986D+01/
      DATA Q1/0.17687158D-02,0.10497312D-02,0.10010797D-02/
      DATA Q2/0.13656599D-03,0.16917282D-03,0.43687862D-03/
      IERR=0
C
C Post-multiply the tristimulus coordinates by the inverse of the
C chromaticity coordinate matrix, to convert the tristimulus
C coordinates to excitations.
C
      E(1)= 1.92908189D0*X-0.54480480D0*Y-0.32490380D0*Z
      E(2)=-0.94447500D0*X+1.76763786D0*Y+0.04507973D0*Z
      E(3)= 0.01539308D0*X-0.22283305D0*Y+1.27982404D0*Z
C
C Use inverse gamma correction (i.e., find the roots of the gamma
C correction equations) to convert the excitations to LUT entries.
```

APPENDIX A (Continued)

```
C
      DO 300 I=1,3
      E0=Q0(I)+V0(I)*(Q1(I)+V0(I)*Q2(I))
      IF(E(I).LE.E0) GOTO 200
C
C Find the roots of a power function.
C
      E1=P2(I)/P3(I)
      E2=P1(I)/P3(I)
      E3=(P0(I)-LOG(E(I)))/P3(I)
      EOFF=E1/3.0D0
      A=(E1*E1-(E2+E2+E2))/9.0D0
      B=(2.0D0*E1*E1*E1-9.0D0*E1*E2+27.0D0*E3)/54.0D0
      IF(B*B.GT.A*A*A) GOTO 100
C
C Three real roots
C
      THETA=ACOS(B/SQRT(A*A*A))
      ETERM=-2.0D0*SQRT(A)
      V(I)=EXP(ETERM*COS((THETA+12.5663706144D0)/3.0D0)-EOFF)
      IF(V(I).LT.(V0(I)-0.1D0).OR.V(I).GT.255.1D0)
     *  V(I)=EXP(ETERM*COS((THETA+6.28318530718D0)/3.0D0)-EOFF)
      IF(V(I).LT.(V0(I)-0.1D0).OR.V(I).GT.255.1D0)
     *  V(I)=EXP(ETERM*COS(THETA/3.0D0)-EOFF)
      IF(V(I).LT.(V0(I)-0.1D0).OR.V(I).GT.255.1D0) IERR=1
      GOTO 300
C
C One real root
C
100   ETERM=EXP(LOG(SQRT(B*B-A*A*A)+ABS(B))/3.0D0)
```

APPENDIX A (Continued)

```
      V(I)=EXP(-SIGN(1.0D0,B)*(ETERM+(A/ETERM))-EOFF)
      IF(V(I).LT.(V0(I)-0.1D0).OR.V(I).GT.255.1D0) IERR=1
      GOTO 300
C
C Find the roots of a quadratic function.
C
200   A=-0.5*(Q1(I)+SIGN(1.0D0,Q1(I))*SQRT(Q1(I)*Q1(I)-
     *  4.0D0*(Q0(I)-E(I))*Q2(I)))
      V(I)=A/Q2(I)
      IF(V(I).LT.-0.1D0.OR.V(I).GT.(V0(I)+0.1D0)) V(I)=(Q0(I)-E(I))/A
      IF(V(I).LT.-0.1D0.OR.V(I).GT.(V0(I)+0.1D0)) IERR=1
300   CONTINUE
      LUTR=V(1)+0.5
      LUTG=V(2)+0.5
      LUTB=V(3)+0.5
      RETURN
      END

      SUBROUTINE GAMMA(IGUN,LUT,E)
C
C FORTRAN subroutine to perform gamma correction (i.e., convert the LUT
C entry LUT--an INTEGER*2 variable--to its corresponding phosphor
C excitation E--a REAL*8 variable).  IGUN is an INTEGER*2 variable that
C equals 1 for red phosphors, 2 for green phosphors, and 3 for blue
C phosphors.  All internal calculations are in double precision.
C
C ** WARNING ** The parameter values in this subroutine should not be
C generalized to other video systems.
C
```

APPENDIX A (Continued)

```
      IMPLICIT REAL*8 (E,V)
      REAL*8 V0(3),P0(3),P1(3),P2(3),P3(3),Q0(3),Q1(3),Q2(3)
      DATA V0/0.11764310D+03,0.14000000D+03,0.10949059D+03/
      DATA P0/0.15284195D+02,-.74795130D+02,0.39596094D+02/
      DATA P1/-.11514202D+02,0.39700333D+02,-.24230622D+02/
      DATA P2/0.25233193D+01,-.71598417D+01,0.48892751D+01/
      DATA P3/-.15663061D+00,0.45336301D+00,-.30539627D+00/
      DATA Q0/0.62875237D-01,0.51273829D-01,0.20635986D+01/
      DATA Q1/0.17687158D-02,0.10497312D-02,0.10010797D-02/
      DATA Q2/0.13656599D-03,0.16917282D-03,0.43687862D-03/
      V=LUT
      IF(V.LE.V0(IGUN)) THEN
        E=Q0(IGUN)+V*(Q1(IGUN)+V*Q2(IGUN))
      ELSE
        VLOG=LOG(V)
        E=EXP(P0(IGUN)+VLOG*(P1(IGUN)+VLOG*(P2(IGUN)+VLOG*P3(IGUN))))
      ENDIF
      RETURN
      END
```

APPENDIX B
Subroutine to Fill an LUT Using a Grid Search

```
      SUBROUTINE GRID(X0,Y0,Z0,XDIR,YDIR,ZDIR,APMAX,LENGTH,LUT)
C
C FORTRAN subroutine to fill an LUT using a grid search.
C
C Input variables (REAL*4):
C
C     X0      = The X-coordinate of the initial color
C     Y0      = The Y-coordinate of the initial color
```

APPENDIX B (Continued)

```
C     Z0      = The Z-coordinate of the initial color
C     XDIR    = The desired relative change in X
C     YDIR    = The desired relative change in Y
C     ZDIR    = The desired relative change in Z
C     APMAX   = The maximum acceptable quantization error
C
C Output variables (INTEGER*2):
C
C     LENGTH = The number of entries written to the LUT.  If
C              LENGTH=0, the starting color cannot be displayed.
C     LUT    = An array, with a minimum size of (3,LENGTH), in which
C              the LUT entries are returned.  The initial LUT entry
C              is returned in LUT(1,1) to LUT(3,1), the next entry is
C              returned in LUT(1,2) to LUT(3,2), and so on.
C
C NOTE: The DIMENSION statement and statement 80 should be changed
C accordingly if the maximum number of LUT entries differs from 256.
C
      REAL*8 ER(21),EG(21),EB(21)
      DIMENSION LUT(3,256)
C
C Initialize LENGTH.
C
      LENGTH=0
C
C Load the initial color into the LUT, returning if it
C cannot be displayed.
C
10    CALL XYZRGB(X0,Y0,Z0,IERR,LUT1,LUT2,LUT3)
```

APPENDIX B (Continued)

```
      IF(IERR.NE.0) RETURN
      LUT(1,1)=LUT1
      LUT(2,1)=LUT2
      LUT(3,1)=LUT3
C
C Calculate DENOM, the denominator for OP.  If DENOM=0,
C return because no movement is desired.
C
      DENOM=SQRT(XDIR*XDIR+YDIR*YDIR+ZDIR*ZDIR)
      LENGTH=1
      IF(DENOM.EQ.0.0) RETURN
C
C Initialize the old value of OP.
C
      OPOLD=0.0
      LENGTH=2
C
C Calculate the excitations, using the following indexing strategy:
C
C   Ec(1) = Excitation for the current LUT entry
C   Ec(2) = Excitation for the current LUT entry, plus 1
C   Ec(3) = Excitation for the current LUT entry, minus 1
C   Ec(4) = Excitation for the current LUT entry, plus 2
C     .
C     .
C     .
C
C where c = R, G, or B (for red, green, and blue).
C
```

APPENDIX B (Continued)

```
20    CALL GAMMA(1,LUT1,ER(1))
      CALL GAMMA(1,LUT1+1,ER(2))
      CALL GAMMA(1,LUT1-1,ER(3))
      CALL GAMMA(2,LUT2,EG(1))
      CALL GAMMA(2,LUT2+1,EG(2))
      CALL GAMMA(2,LUT2-1,EG(3))
      CALL GAMMA(3,LUT3,EB(1))
      CALL GAMMA(3,LUT3+1,EB(2))
      CALL GAMMA(3,LUT3-1,EB(3))
C
C Initialize the grid depth (NGRID) and the size of the previous
C grid (NPREV).
C
      NGRID=1
      NPREV=1
C
C Search a new grid, or expand the current grid.  NMAX is the grid size
C OPMIN is the smallest value of OP found so far, and APMIN is the
C smallest value of AP found so far.
C
30    NMAX=2*NGRID+1
      OPMIN=999.9
      APMIN=999.9
C
C Combine the excitations factorially, ignoring LUT entries that
C exceed the bounds of the apparatus.
C
      DO 70 I=1,NMAX
      IADJ=I/2
```

APPENDIX B (Continued)

```
      LUTR=LUT1+(2*(IADJ+IADJ-I)+1)*IADJ
      IF(LUTR.LT.0.OR.LUTR.GT.255) GOTO 70
      DO 60 J=1,NMAX
      JADJ=J/2
      LUTG=LUT2+(2*(JADJ+JADJ-J)+1)*JADJ
      IF(LUTG.LT.0.OR.LUTG.GT.255) GOTO 60
      DO 50 K=1,NMAX
C
C Ignore combinations that have already been examined.
C
        IF(I.LE.NPREV.AND.J.LE.NPREV.AND.K.LE.NPREV) GOTO 50
        KADJ=K/2
        LUTB=LUT3+(2*(KADJ+KADJ-K)+1)*KADJ
        IF(LUTB.LT.0.OR.LUTB.GT.255) GOTO 50
C
C Post-multiply the excitations by the chromaticity coordinates, to
C convert them to tristimulus coordinates.
C
C ** WARNING ** The chromaticity coordinates listed below are specific
C to a Barco CDCT 5137 monitor, and should not be generalized to other
C monitors.
C
        X=0.62D0*ER(I)+0.210D0*EG(J)+0.15D0*EB(K)
        Y=0.33D0*ER(I)+0.675D0*EG(J)+0.06D0*EB(K)
        Z=0.05D0*ER(I)+0.115D0*EG(J)+0.79D0*EB(K)
C
C Calculate OP for the candidate.
C
        DX=X-X0
```

APPENDIX B (Continued)

```
      DY=Y-Y0
      DZ=Z-Z0
      OP=(DX*XDIR+DY*YDIR+DZ*ZDIR)/DENOM
C
C Reject the candidate if OP did not increase.
C
      IF(OP.LE.OPOLD) GOTO 50
C
C Reject the candidate if another acceptable candidate produced a
C smaller change in OP.
C
      IF(OP.GT.OPMIN) GOTO 50
C
C Reject the candidate if AP is too large.
C
      AP=SQRT(DX*DX+DY*DY+DZ*DZ-OP*OP)
      IF(AP.GT.APMAX) GOTO 50
C
C If two candidates produce the same change in OP, keep the one
C with the smallest error.
C
      IF(OP.LT.OPMIN) GOTO 40
      IF(AP.GE.APMIN) GOTO 50
C
C Retain the current candidate, which is the best so far.
C
40    LUT(1,LENGTH)=LUTR
      LUT(2,LENGTH)=LUTG
      LUT(3,LENGTH)=LUTB
```

APPENDIX B (Continued)

```
      OPMIN=OP
      APMIN=AP
50    CONTINUE
60    CONTINUE
70    CONTINUE
C
C Check whether a new LUT entry was found (in which case OPMIN will
C be less than its initial value).  If not, expand the grid or--if
C the grid is already at its maximum size--exit from the routine.
C
      IF(OPMIN.LT.999.9) GOTO 80
      IF(NGRID.EQ.10) GOTO 90
      NPREV=NMAX
      NGRID=NGRID+1
      CALL GAMMA(1,LUT1+NGRID,ER(NMAX+1))
      CALL GAMMA(1,LUT1-NGRID,ER(NMAX+2))
      CALL GAMMA(2,LUT2+NGRID,EG(NMAX+1))
      CALL GAMMA(2,LUT2-NGRID,EG(NMAX+2))
      CALL GAMMA(3,LUT3+NGRID,EB(NMAX+1))
      CALL GAMMA(3,LUT3-NGRID,EB(NMAX+2))
      GOTO 30
C
C Exit from the routine if the LUT is full.
C
80    IF(LENGTH.EQ.256) RETURN
C
C Update the old value of OP and search for the next entry.
C
      OPOLD=OPMIN
```

APPENDIX B (Continued)

```
      LUT1=LUT(1,LENGTH)
      LUT2=LUT(2,LENGTH)
      LUT3=LUT(3,LENGTH)
      LENGTH=LENGTH+1
      GOTO 20
C
C GRID only gets here if it was looking for an LUT entry, but
C couldn't find one.  Adjust the value of LENGTH before returning.
C
90    LENGTH=LENGTH-1
      RETURN
      END
```

APPENDIX C
Subroutine to Fill an LUT Using Vector Addition

```
      SUBROUTINE ADDM(X0,Y0,Z0,XDIR,YDIR,ZDIR,APMAX,LENGTH,LUT)
C
C FORTRAN subroutine to fill an LUT using vector addition.
C Input and output variables are the same as for subroutine GRID.
C
C NOTE: The DIMENSION statement and statement 105 should be changed
C accordingly if the maximum number of LUT entries differs from 256.
C
      DIMENSION LUT(3,256),OP(256)
C
C Initialize LENGTH.
C
      LENGTH=0
C
```

APPENDIX C (Continued)

```
C Load the initial color into the LUT, returning if it
C cannot be displayed.
C
10    CALL XYZRGB(X0,Y0,Z0,IERR,LUT1,LUT2,LUT3)
      IF(IERR.NE.0) RETURN
      LUT(1,1)=LUT1
      LUT(2,1)=LUT2
      LUT(3,1)=LUT3
      LENGTH=1
C
C Rescale the entries in the movement vector so that the most
C extreme entry has an absolute value of unity.  Return if no
C movement is desired.
C
      DELTAX=XDIR
      DELTAY=YDIR
      DELTAZ=ZDIR
      SCALE=ABS(DELTAX)
      IF(ABS(DELTAY).GT.SCALE) SCALE=ABS(DELTAY)
      IF(ABS(DELTAZ).GT.SCALE) SCALE=ABS(DELTAZ)
      IF(SCALE.LE.0.0) RETURN
      DELTAX=DELTAX/SCALE
      DELTAY=DELTAY/SCALE
      DELTAZ=DELTAZ/SCALE
C
C Initialize the array containing values of OP, and the
C denominator for calculating OP.
C
      OP(1)=0.0
```

APPENDIX C (Continued)

```
      DENOM=SQRT(XDIR*XDIR+YDIR*YDIR+ZDIR*ZDIR)
C
C Set the current color to the initial color.
C
      X=X0
      Y=Y0
      Z=Z0
C
C Add the movement vector for the first time, and find the
C corresponding LUT entry.
C
20    CALL XYZRGB(X+DELTAX,Y+DELTAY,Z+DELTAZ,IERR,LUTR,LUTG,LUTB)
C
C Check if the LUT entry changed.  If it did not, add the movement
C vector a second time.  Otherwise, scale the movement vector down.
C There is no need to check IERR, because one of the LUT entries
C will have changed if IERR is not 0.
C
      IF(LUT1.NE.LUTR) GOTO 30
      IF(LUT2.NE.LUTG) GOTO 30
      IF(LUT3.EQ.LUTB) GOTO 40
C
C Scale the movement vector down.
C
30    DELTAX=0.5*DELTAX
      DELTAY=0.5*DELTAY
      DELTAZ=0.5*DELTAZ
      GOTO 20
C
```

APPENDIX C (Continued)

```
C Add the movement vector a second time, and find the
C corresponding LUT entry.
C
40     XDESIR=X+DELTAX+DELTAX
       YDESIR=Y+DELTAY+DELTAY
       ZDESIR=Z+DELTAZ+DELTAZ
       CALL XYZRGB(XDESIR,YDESIR,ZDESIR,IERR,LUTR,LUTG,LUTB)
C
C Check whether the LUT entry changed.
C
       IF(LUT1.NE.LUTR) GOTO 50
       IF(LUT2.NE.LUTG) GOTO 50
       IF(LUT3.NE.LUTB) GOTO 50
C
C The LUT did not change after 2 additions of the movement vector.
C Scale the movement vector up and proceed directly to the second
C addition (since it is already known that the first addition will
C not change the LUT).
C
       DELTAX=DELTAX+DELTAX
       DELTAY=DELTAY+DELTAY
       DELTAZ=DELTAZ+DELTAZ
       GOTO 40
C
C The LUT entry changed.  Return if the new entry exceeds the
C bounds of the apparatus.
C
50     IF(IERR.NE.0) RETURN
C
C The current LUT entry should be saved.  Determine where to insert
```

APPENDIX C (Continued)

```
C it into the sequence of previous entries so that OP increases.
C
      CALL RGBXYZ(LUTR,LUTG,LUTB,X1,Y1,Z1)
      DX=X1-X0
      DY=Y1-Y0
      DZ=Z1-Z0
      OPTEMP=(DX*XDIR+DY*YDIR+DZ*ZDIR)/DENOM
      AP=SQRT(DX*DX+DY*DY+DZ*DZ-OPTEMP*OPTEMP)
      IF(AP.GT.APMAX) GOTO 110
      I=LENGTH
60    IF(OPTEMP.GT.OP(I)) GOTO 80
C
C If OP is exactly the same as for a previous entry, keep the
C entry with the smallest quantization error.
C
      IF(OPTEMP.NE.OP(I)) GOTO 70
      CALL RGBXYZ(LUT(1,I),LUT(2,I),LUT(3,I),X2,Y2,Z2)
      DX=X2-X0
      DY=Y2-Y0
      DZ=Z2-Z0
      AP2=SQRT(DX*DX+DY*DY+DZ*DZ-OP(I)*OP(I))
      IF(AP2.LE.AP) GOTO 110
      LUT(1,I)=LUTR
      LUT(2,I)=LUTG
      LUT(3,I)=LUTB
      GOTO 110
```

APPENDIX C (Continued)

```
C
C If OP is shorter than for all previous entries, reject the
C current entry because OP must be negative.
C
70      IF(I.EQ.1) GOTO 110
        I=I-1
        GOTO 60
C
C Make room for the current entry by advancing the previous
C entries as needed.
C
80      I=I+1
        LENGTH=LENGTH+1
        IF(I.EQ.LENGTH) GOTO 100
        DO 90 J=LENGTH,I+1,-1
        LUT(1,J)=LUT(1,J-1)
        LUT(2,J)=LUT(2,J-1)
        LUT(3,J)=LUT(3,J-1)
90      OP(J)=OP(J-1)
C
C Save the current entry.
C
100     LUT(1,I)=LUTR
        LUT(2,I)=LUTG
        LUT(3,I)=LUTB
        OP(I)=OPTEMP
C
C Exit from the routine if the LUT is full.
C
```

APPENDIX C (Continued)

```
105   IF(LENGTH.EQ.256) RETURN
110   LUT1=LUTR
      LUT2=LUTG
      LUT3=LUTB
C
C Update the desired coordinates and find the next entry.
C
      X=XDESIR
      Y=YDESIR
      Z=ZDESIR
      GOTO 20
      END
```

(Manuscript received February 22, 1989;
revision accepted for publication August 8, 1989.)