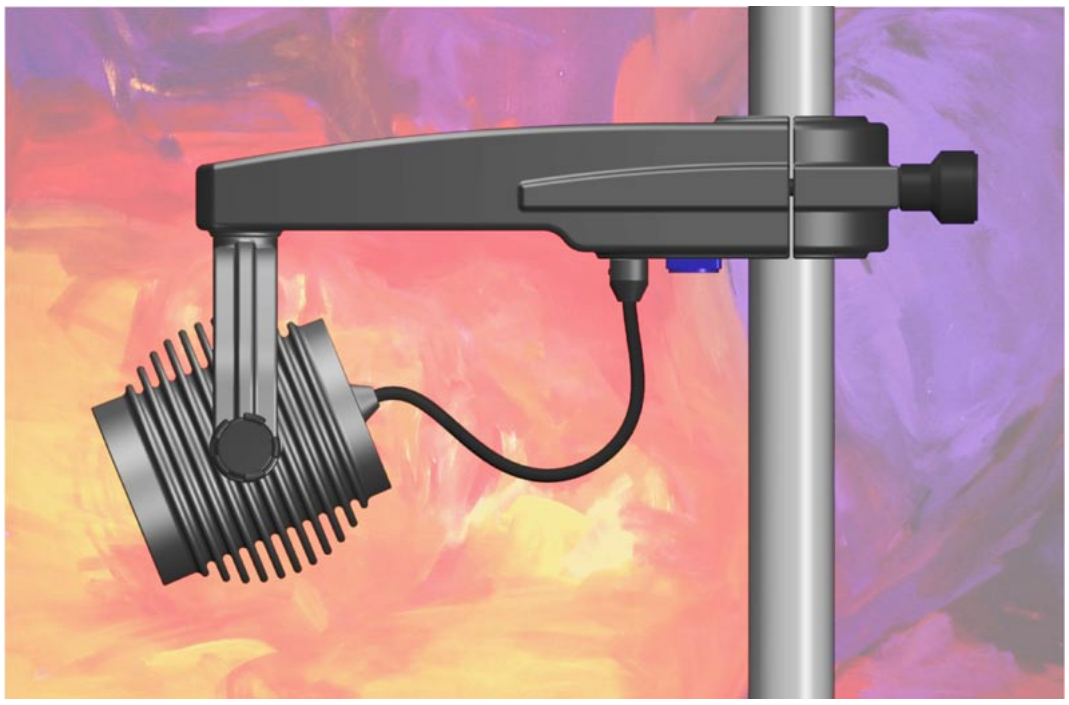
Gernot Hoffmann

Gamut for CIE and sRGB Primaries with Varying Luminance

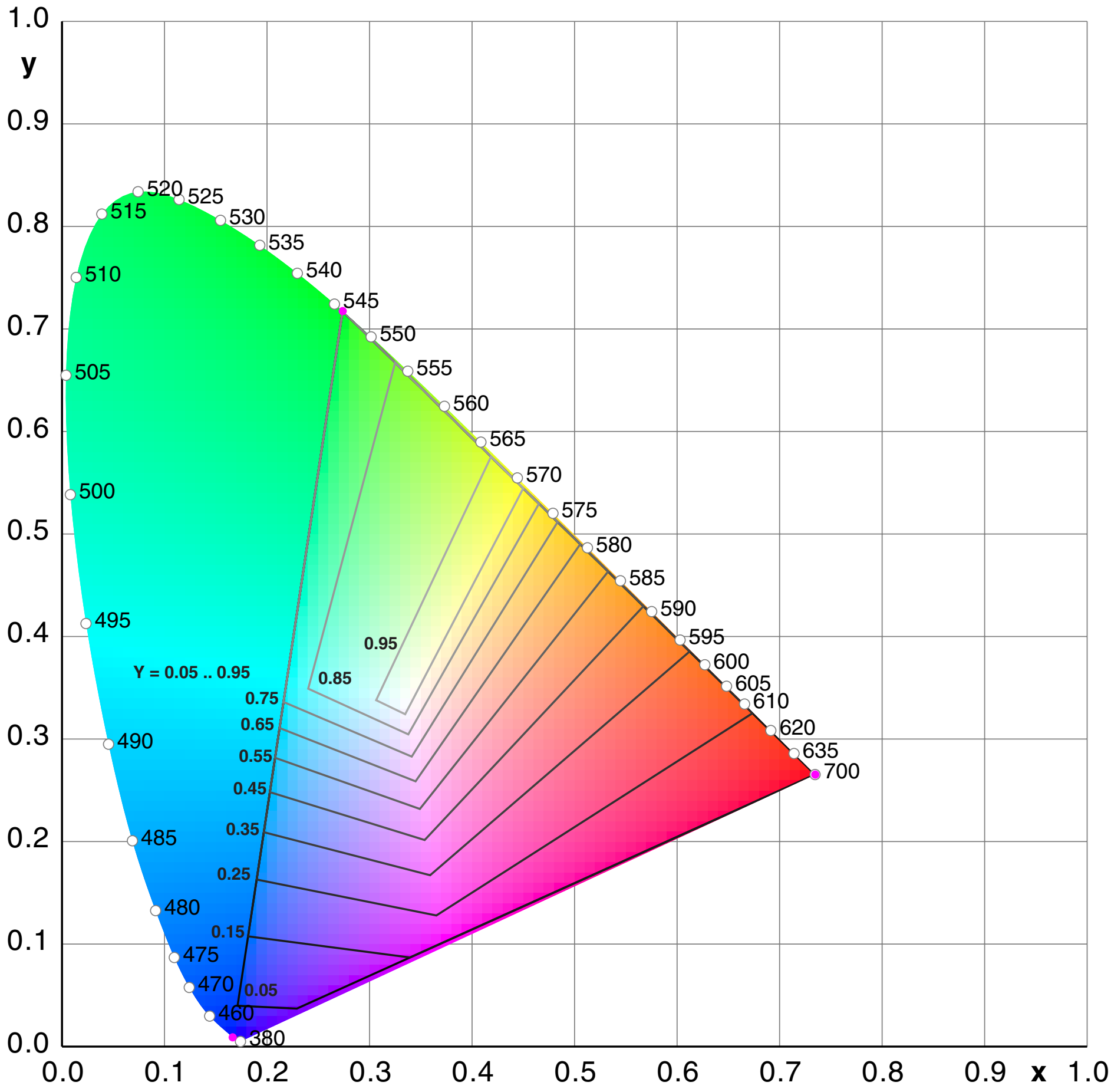


Contents

1. Introduction 2
2. Mathematics for Gamut Boundaries 3
3. Sorting the Gamut Corners 4
4. PostScript Code 5
5. Rec. 709 Primaries (sRGB) 12
6. References 13

1. Introduction

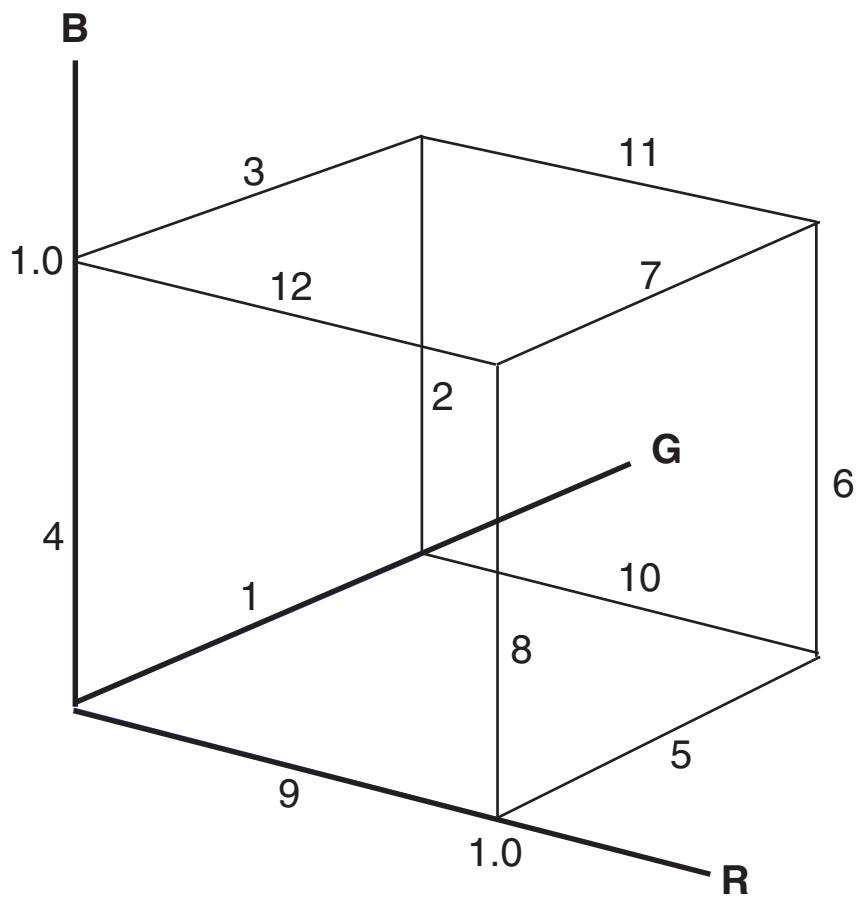
The gamut for any RGB color order system is a triangle in the CIE xyY chromaticity diagram, here shown for the CIE primaries, indicated by magenta dots. This gamut is the boundary by projecting all colors from 3D XYZ onto the 2D xy plane. The actual gamut depends on the luminance.
Luminance is the Y-value, and Y=1 delivers always R=G=B=1.
E.g. for Y=0.95 the gamut seems to be very small, a flat rectangle. This means, that outside the rectangle at least one color C of RGB is clipped for the range $0 \leq C \leq 1$, using normalized values instead of 0 to 255.
Colors out of gamut are not correctly reproduced though they may still look reasonable.

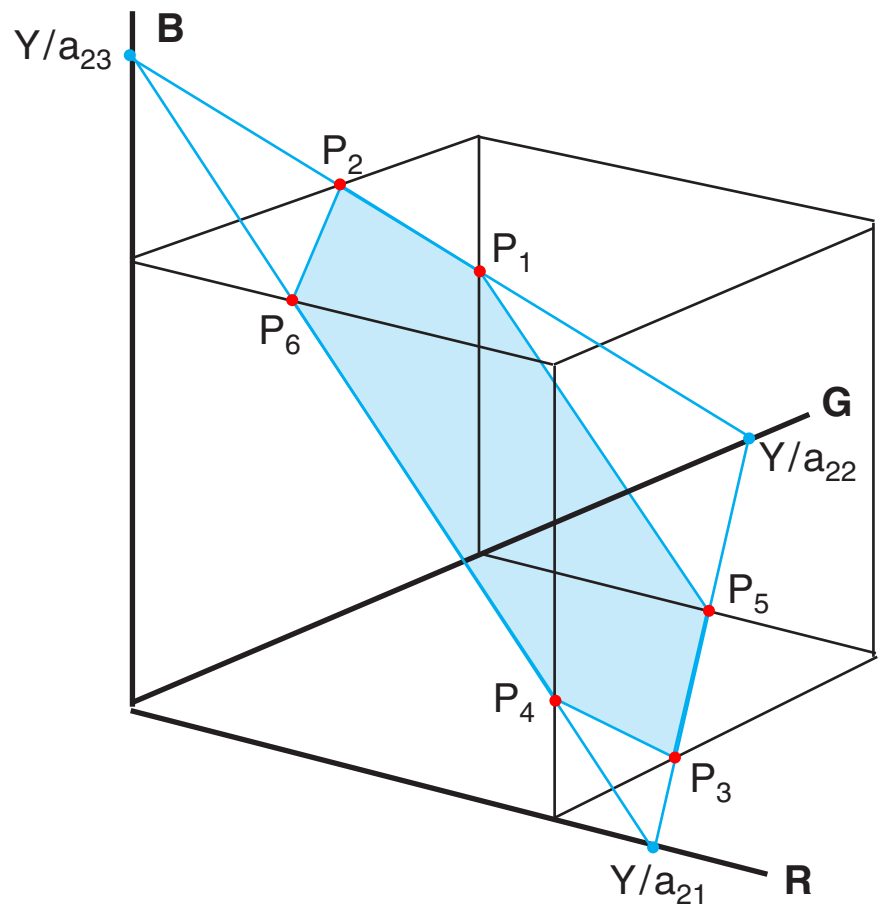


2. Mathematics for Gamut Boundaries

The transformation between CIE XYZ and a linear RGB space (no Gamma encoding) is merely a matrix multiplication.

(1) $X = a_{11} R + a_{12} G + a_{13} B$
(2) $Y = a_{21} R + a_{22} G + a_{23} B$
(3) $Z = a_{31} R + a_{32} G + a_{33} B$





For a given value Y the second equation delivers a plane in RGB, marked by blue lines. The limitation for R,G,B in the range 0 to 1 is represented by a cube, shown by black lines. The plane intersects for 0<Y<1 the cube and delivers a polygon with three to six corners. The illustration shows a hexagon.
Now we have to find all intersections P_1 to P_n with all the 12 edges. Check these cases:

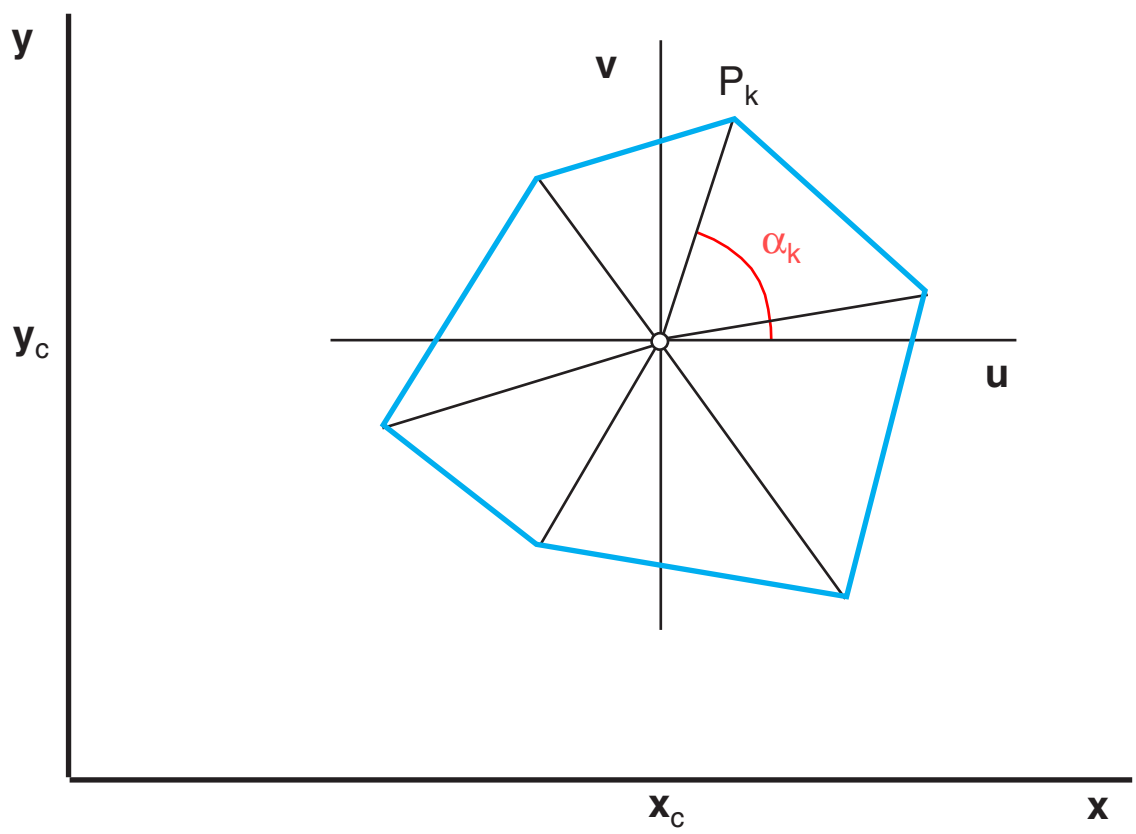
1	R =0	B =0	G	0 <	(Y/a_{22})	< 1
2	R =0	G =1	B	0 <	$(Y-a_{22})/a_{23}$	< 1
3	R =0	B =1	G	0 <	$(Y-a_{23})/a_{22}$	< 1
4	R =0	G =0	B	0 <	(Y/a_{23})	< 1
5	R =1	B =0	G	0 <	$(Y-a_{21})/a_{22}$	< 1
6	R =1	G =1	B	0 <	$(Y-a_{21}-a_{22})/a_{23}$	< 1
7	R =1	B =1	G	0 <	$(Y-a_{21}-a_{23})/a_{22}$	< 1
8	R =1	G =0	B	0 <	$(Y-a_{21})/a_{23}$	< 1
9	G =0	B =0	R	0 <	(Y/a_{21})	< 1
10	G =1	B =0	R	0 <	$(Y-a_{22})/a_{21}$	< 1
11	G =1	B =1	R	0 <	$(Y-a_{22}-a_{23})/a_{21}$	< 1
12	G =0	B =1	R	0 <	$(Y-a_{23})/a_{21}$	< 1

3. Sorting the Gamut Corners

Once we have found n=3 to 6 corners of the polygon in RGB, we can transform them into CIE xyY. First we apply Eq. (1) and (3) and then we map to xy:

$x = X/(X+Y+Z)$
$y = Y/(X+Y+Z)$

As we can see in the illustration on the previous page, the corners are not necessarily sorted. Therefore we have to sort them counterclockwise before they are connected by lines.



The center of the polygon is found by the mean values of the x- and the y-coordinates. Then the corners are transformed into the local uv-coordinate system.
Each corner has an angle α_k which is found by a four-quadrant function α =arctan(v/u). This is not an ordinary arctan function.

In PostScript we use *v u atan* and the result α is in the range 0° to 360°.
In C we use α *=atan2(v,u)* and the result α is in the range $-\pi$ to $+\pi$.

Then the list of n=3 to 6 sets (u,v,α) is sorted for increasing α, though this would not be necessary for n=3 .

Everything else can be taken from the PostScript source file.

The sorting could be done in 3D RGB. The rotation axis is the normal vector and the center is found by averaging three coordinates for n points.

4.1 PostScript Code

```
%!PS-Adobe-3.0    EPSF-3.0
%%BoundingBox:    0 0 404 340
%%Creator:        Gernot Hoffmann
%%Title:          Cie Gamut
%%CreationDate:   Jan.14,2003
%%DocumentNeededResources: font Helvetica
%%+ Helvetica-Bold

% Disable setpagedevice/setpagedevice {pop} bind def

%-Variables

/mm {2.834646 mul} def % points per mm

/ci [ 380 0.1741 0.0050
      385 0.1740 0.0050
      390 0.1738 0.0049
      395 0.1736 0.0049
      400 0.1733 0.0048
      405 0.1730 0.0048
      410 0.1726 0.0048
      415 0.1721 0.0048
      420 0.1714 0.0051
      425 0.1703 0.0058
      430 0.1689 0.0069
      435 0.1669 0.0086
      440 0.1644 0.0109
      445 0.1611 0.0138
      450 0.1566 0.0177
      455 0.1510 0.0227
      460 0.1440 0.0297
      465 0.1355 0.0399
      470 0.1241 0.0578
      475 0.1096 0.0868
      480 0.0913 0.1327
      485 0.0686 0.2007
      490 0.0454 0.2950
      495 0.0235 0.4127
      500 0.0082 0.5384
      505 0.0039 0.6548
      510 0.0139 0.7502
      515 0.0389 0.8120
      520 0.0743 0.8338
      525 0.1142 0.8262
      530 0.1547 0.8059
      535 0.1929 0.7816
      540 0.2296 0.7543
      545 0.2658 0.7243
      550 0.3016 0.6923
      555 0.3373 0.6589
      560 0.3731 0.6245
      565 0.4087 0.5896
      570 0.4441 0.5547
      575 0.4788 0.5202
      580 0.5125 0.4866
      585 0.5448 0.4544
      590 0.5752 0.4242
      595 0.6029 0.3965
      600 0.6270 0.3725
      605 0.6482 0.3514
      610 0.6658 0.3340
      615 0.6801 0.3197
      620 0.6915 0.3083
      625 0.7006 0.2993
      630 0.7079 0.2920
      635 0.7140 0.2859
      640 0.7190 0.2809
      645 0.7230 0.2770
      650 0.7260 0.2740
      655 0.7283 0.2717
      660 0.7300 0.2700
      665 0.7311 0.2689
      670 0.7320 0.2680
      675 0.7327 0.2673
      680 0.7334 0.2666
      685 0.7340 0.2660
      690 0.7344 0.2656
      695 0.7346 0.2654
      700 0.7347 0.2653 ] def

/ck [380 460 470 475 480 485 490 495 500
      505 510 515 520 525 530 535 540 545
      550 555 560 565 570 575 580 585 590
      595 600 605 610 620 635 700 ] def

/cp [435.8  0.1665 0.0090
      546.1  0.2738 0.7173
      700.0  0.7347 0.2653 ] def

/RGB 27 array def
/xyY 27 array def

/sx 100 mm def
/sy 100 mm def
/bx 140 mm def
/by 120 mm def

/x0  10 mm def
/y0  10 mm def
```

4.2 PostScript Code

```
/xa  x0 def
/xe  x0 sx add def
/ya  y0 def
/ye  y0 sy add def

/gam 1 2.2 div def

%-Procedures

/Grid
{ newpath
  0.5 setgray
  0.1 mm sx div setlinewidth
 /gx 0 def  /gdx 0.1 def
 /gy 0 def  /gdy 0.1 def
 { gx gy moveto
      gx 1 add gy lineto stroke
  /gy gy gdy add def gy 1.01 gt {exit}if } loop
 /gy 0 def
 { gx gy moveto
      gx gy 1 add lineto stroke
  /gx gx gdx add def gx 1.01 gt {exit}if } loop
0 setlinecap
%—axes
newpath
0 0 moveto 1 0 lineto 0 0 moveto 0 1 lineto
0.2 mm sx div setlinewidth
0.0 setgray
stroke
} def

/MakeSpline
{/fu  spa array def /mi spa array def /re spa array def /yy spa array def /xx spa array def
 /n fp 1 sub def /hh mult def /hd6 hh 6 div def /hm6 hh 6 mul def /h6h -6 hh dup mul div def
  pxy 1 eq {0 1 fp {/k exch def fu k hh mul ci k 3 mul 1 add get put} for} if
  pxy 2 eq {0 1 fp {/k exch def fu k hh mul ci k 3 mul 2 add get put} for} if
  mi 1 0.25 put
  1 1 n 1 sub {/j exch def mi j 1 add 1 4 mi j get sub div put} for
  1 1 n {/j exch def /jh j hh mul def re j fu jh hh add get fu jh get 2 mul sub fu jh hh sub get add h6h mul put} for
  yy 1 re 1 get put
  2 1 n {/j exch def yy j re j get yy j 1 sub get mi j 1 sub get mul sub put} for
  xx 0 0 put xx n yy n get mi n get mul neg put xx n 1 add 0 put
  n 1 sub -1 1{/j exch def xx j yy j get xx j 1 add get add mi j get mul neg put} for
  /i 0 def
  0 1 n
  {/j exch def /jh j hh mul def
      /a3 xx j 1 add get xx j get sub hm6 div def /a2 xx j get 0.5 mul def
      /a1 fu jh hh add get fu jh get sub hh div xx j 1 add get xx j get 2 mul add hd6 mul sub def /a0 fu jh get def
      1 1 hh 1 sub {/k exch def /i i 1 add def fu i a3 k mul a2 add k mul a1 add k mul a0 add put} for
      /i i 1 add def} for
  pxy 1 eq {fu vecx copy} if pxy 2 eq {fu vecy copy} if
} def

/CieHorse
{/mult 5 def
 /fp  ci length 3 idiv 1 sub def
 /spl  fp mult mul def
 /spa  spl 1 add def
 /vecx spa array def
 /vecy spa array def
  newpath
 /pxy 1 def MakeSpline
 /pxy 2 def MakeSpline
  vecx 0 get vecy 0 get moveto
  1 1 spl
  {/k exch def
      vecx k get vecy k get lineto } for
  closepath
  clip
} def

/R 0 def /G 0 def /B 0 def /z 0 def /X 0 def /Z 0 def /max 0 def

/XyzRgb
{ /z 1 x sub y sub def
  /X Y x mul y div def
  /Z Y z mul y div def
  /R X      2.3647 mul Y 0.8965 mul sub Z 0.4681 mul sub def
  /G X neg 0.5152 mul Y 1.4264 mul add Z 0.0888 mul add def
  /B X      0.0052 mul Y 0.0144 mul sub Z 1.0092 mul add def
  /max R def
      G max gt {/max G def } if
      B max gt {/max B def } if
  /R R max div def
  /G G max div def
  /B B max div def
      R 0 lt {/R 0 def } if
      G 0 lt {/G 0 def } if
      B 0 lt {/B 0 def } if
  /R R gam exp def
  /G G gam exp def
  /B B gam exp def
      R G B setrgbcolor
} bind def
```

4.3 PostScript Code

```
/CieArea
{ newpath
 /n 100 def % divisions of color patches
 /e 1 n div def
 /Y 1.0 def
 /y 0.000001 def
  1 1 n 0.85 mul round
{ pop
 /x 0 def
  1 1 n 1 y sub mul round
{ pop
  XyzRgb
  x y  moveto e 0 rlineto 0 e rlineto e neg 0 rlineto
  closepath fill
 /x x e add def } for
 /y y e add def } for
} def

/CieDots % draw circle for ck lambdas
{ /col { 0.5 setgray } def
  0.1 mm sx div setlinewidth
 /ra 0.5 mm sx div def
 /k 0 def
 /i 0 def
  1 1 ck length
 {  pop
      /lam ck k get def
      { ci i get lam eq
      { /x ci i 1 add get def
      /y ci i 2 add get def
      x y ra 0 360 arc
      gsave 1 setgray fill grestore col stroke
      exit }
      {/i i 3 add def } ifelse      } loop
  /k k 1 add def
 } for
} def

/CieList % write list for ck lambdas
{ 0.3 0.3 1 setrgbcolor
 /fh 9 sx div def
 /Helvetica findfont fh scalefont setfont
 /txt 3 string def
 /x 1.1 def
 /y 1.0 fh sub def
 /dy fh 1.2 mul def
 /k 0 def
 /i 0 def
  1 1 ck length
 { pop
      /lam ck k get def
      { ci i get lam eq
      { x y moveto
      lam txt cvs txt show
      /y y dy sub def
      exit }
      {/i i 3 add def } ifelse      } loop
  /k k 1 add def
 } for
} def

/CieNote % write notation for ck lambdas
{ /col { 0 setgray } def
 /fh 6 sx div def
 /Helvetica findfont fh scalefont setfont
 /txt 3 string def
 /dx fh 2.0 mul def
 /dy fh def
 /k 0 def
 /i 0 def
  1 1 ck length
 { pop
      /lam ck k get round def
      { ci i get round lam eq
      { /x ci i 1 add get fh 0.40 mul add def
      /y ci i 2 add get fh 0.18 mul sub def
      newpath
      x fh 0.1 mul sub y 0.15 fh mul sub moveto
      dx 0 rlineto
      0 dy rlineto
      dx neg 0 rlineto
      closepath
      1 setgray
      % fill
      col
      x y moveto
      lam txt cvs txt show
      exit }
      {/i i 3 add def } ifelse      } loop
  /k k 1 add def
 } for
} def
```

4.4 PostScript Code

```
/CiePrim % primaries cp
{/col { 1 0 1 setrgbcolor } def
 /ra 0.4 mm sx div def
 /k 0 def
  1 1 3
 {  pop
  /x  cp k 1 add get def /y cp k 2 add get def
      x y ra 0 360 arc
      col fill
  /k k 3 add def } for
} def

/Shownum
% Draw number by string
%  Global
%  xtxt     Actual position not overwritten
%  ytxt
%  nu Input number      not overwritten  nu =+-999999
%  fh Font height not overwritten
%  tms      Mantissa number of characters tms=0...6
%  tms=3  Example
%  input  -23.56789  -999.99  0.4567  9999.123456
%  result -23.568 -999.990    0.467 9999.123
%  Postscript number to string is not well defined
%  e.g. 1E-5 instead of 0.00001
%  We use a straightforward BCD conversion.
%  This is always affected by round-off errors
%  because of 32-bit arithmetic
%  The procedure itself does not round
%  Results are different, depending on the interpreter

{ /tx0 xtxt def
  /ty0 ytxt def
  /tfw fh 0.6 mul def   % character distance
  /tna nu abs 0.5E-6 add def  % abs value

  /tdec 1E5 def
  /tchr 1 string def

      tna 999999.1 lt   % larger number replaced by #
  /tmm true def   % sign
  {/tx0 tx0 tfw 6 mul sub def
      /tz 0 def
  1 1 5     % first 5 digits, no leading 0
  { pop
      /tk 0 def
      { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
      } loop
      tk 0 ne {/tz tz 1 add def} if
      tz 0 ne
      { tx0 ty0 moveto tk tchr cvs show
      } if
      tz 1 eq nu 0 lt and     % minus
      { tx0 tfw 0.7 mul sub ty0 moveto (-) show
      /tmm false def
      } if
      /tdec tdec 0.1 mul def
      /tx0 tx0 tfw add def
  } for

      /tk 0 def   % leading 0
      { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
      } loop
      tmm nu 0 lt and   % minus
      { tx0 tfw 0.7 mul sub ty0 moveto (-) show
      } if
      tx0 ty0 moveto tk tchr cvs show
      /tdec tdec 0.1 mul def
      /tx0 tx0 tfw add def

  tms 0 gt  % for float
  { tx0 ty0 moveto (.) show
      /tx0 tx0 tfw 0.5 mul add def
  1 1 tms
  { pop
      /tk 0 def
      { tna tdec gt {/tna tna tdec sub def /tk tk 1 add def}{exit} ifelse
      } loop
      tx0 ty0 moveto tk tchr cvs show
      /tdec tdec 0.1 mul def
      /tx0 tx0 tfw add def
  } for
  } if
 }{ tx0 tfw sub ty0 moveto (#) show} ifelse
} def
```

4.5 PostScript Code

```
/CieText
{/num [0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 ] def
 /tx  3 string def
 /fh  8 sx div def
 /Helvetica findfont fh scalefont setfont
 /xtxt fh -0.1 mul def
 /ytxt fh 1.2 mul neg def
 /tms 1 def
  0 setgray
  0 1 10
 {/k exch def
  /nu num k get def
      Shownum
  /xtxt xtxt 0.1 add def
 } for
 /xtxt fh 1.3 mul neg def
 /ytxt fh 0.2 mul neg def
 0 1 10
 {/k exch def
  /nu num k get def
      Shownum
  /ytxt ytxt 0.1 add def
 } for
 /Helvetica-Bold findfont fh scalefont setfont
  0.945 fh 1.2 mul neg moveto (x) show
 -0.04 0.95 moveto (y) show
 } def


/RGBxyY
{ /X a11 R mul a12 G mul add a13 B mul add def
  /Z a31 R mul a32 G mul add a33 B mul add def
  /S X Y add Z add def
  /x X S div def
  /y Y S div def
} def


/CieGamut
{ % X = A*R
  /a11 0.4900 def /a12 0.3100 def /a13 0.2000 def
  /a21 0.1770 def /a22 0.8124 def /a23 0.0106 def
  /a31 0.0000 def /a32 0.0100 def /a33 0.9900 def
  /Y 0.2 def
      1 1 1
 { pop
      Y setgray
      0.5 sx div setlinewidth
  /k  1 def
  /np 0 def
  % 1
  /R 0 def
  /G Y a22 div def
  /B 0 def
      G 1 lt 0 G lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
  % 2
  /R 0 def
  /G 1 def
  /B Y a22 sub a23 div def
      B 1 lt 0 B lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
  % 3
  /R 0 def
  /G Y a23 sub a22 div def
  /B 1 def
      G 1 lt 0 G lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
  % 4
  /R 0 def
  /G 0 def
  /B Y a23 div def
      B 1 lt 0 B lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
  % 5
  /R 1 def
  /G Y a21 sub a22 div def
  /B 0 def
      G 1 lt 0 G lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
  % 6
  /R 1 def
  /G 1 def
  /B Y a21 sub a22 sub a23 div def
      B 1 lt 0 B lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
  % 7
  /R 1 def
  /G Y a21 sub a23 sub a22 div def
  /B 1 def
      G 1 lt 0 G lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
  % 8
  /R 1 def
  /G 0 def
  /B Y a21 sub a23 div def
      B 1 lt 0 B gt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
  % 9
  /R Y a21 div def
  /G 0 def
  /B 0 def
      R 1 lt 0 R lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
```

4.6 PostScript Code

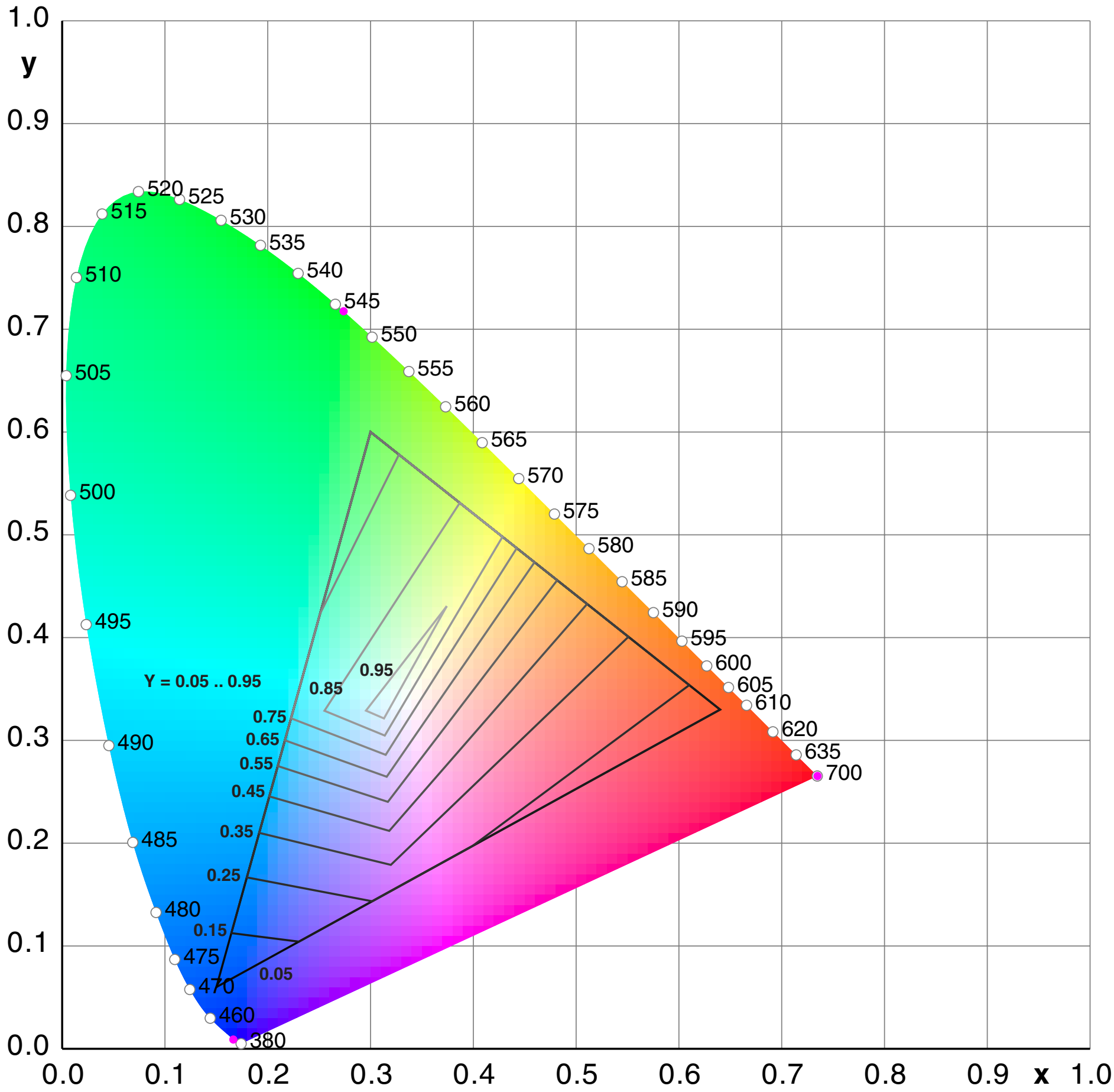
```
 % 10
 /R Y a22 sub a21 div def
 /G 1 def
 /B 0 def
     R 1 lt 0 R lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
 % 11
 /R Y a22 sub a23 sub a21 div def
 /G 1 def
 /B 1 def
     R 1 lt 0 R lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if
 % 12
 /R Y a23 sub a21 div def
 /G 0 def
 /B 1 def
     R 1 lt 0 R lt and { RGB k R put RGB k 1 add G put RGB k 2 add B put /np np 1 add def /k k 3 add def } if

% map to xy
 /k 1 def
 1 1 np
 { pop
     /R RGB k get def /G RGB k 1 add get def /B RGB k 2 add get def
     RGBxyY
     xyY k x put  xyY k 1 add y put
     /k k 3 add def
 } for

% mean value
 /k 1 def
 /xm 0 def
 /ym 0 def
 1 1 np
 { pop
     /xm xm xyY k      get def
     /ym ym xyY k 1 add get def
     /k k 3 add
 } for
 /xm xm np div def
 /ym ym np div def

% shift and angle
 /k 1 def
 1 1 np
 { pop
     xyY k xyY k get xm sub put
     xyY k 1 add xyY k 1 add get ym sub put
     xyY k 2 add xyY k 1 add get xyY k get atan put
     /k k 3 add def
 } for

% sort by angle ccw
 /i 1 def
     1 1 np 1 sub
     { pop
     /ii i 1 sub 3 mul def
     /j  i 1 add def
     /an1 xyY ii 3 add get def
     i 1 add 1 np 1 sub
     { pop
     /jj j 1 sub 3 mul def
     /an2 xyY jj 3 add get def
     an2 an1 lt
     { /px xyY ii      get def
     /py xyY ii 1 add get def
     /pa xyY ii 2 add get def
     xyY ii      xyY jj      get put
     xyY ii 1 add xyY jj 1 add get put
     xyY ii 2 add xyY jj 2 add get put
     xyY jj      px put
     xyY jj 1 add py put
     xyY jj 2 add pa put
     /an1 an2 def      } if
     /j j 1 add def
     } for
     /i i 1 add def
  } for


  /k 1 def
  xyY k get xm add xyY k 1 add get ym add moveto
  2 1 np
  { pop
      /k k 3 add def
      xyY k get xm add xyY k 1 add get ym add lineto
  } for
  closepath
  stroke
  /Y Y 0.1 add def
 } for
} def
```

4.7 PostScript Code

```
%-Begin

true setstrokeadjust
gsave

x0 y0 translate
sx sy scale

Grid
CieHorse
CieArea
CieGamut


grestore
x0 y0 translate
sx sy scale

%CieDots
 CiePrim
%CieNote
 CieText
%CieList

showpage
%%EndDocument
```

5. Rec.709 Primaries (sRGB)

The diagram (similar to page1) for Rec.709 primaries, which are used by sRGB.
Calibrated monitors are near to sRGB.



6. References

[1] PostScript Language Reference
Addison-Wesley
Boston, San Francisco ...
2002

[2] PostScript Language Tutorial and Cookbook
Addison-Wesley
Reading, Massachusetts ...
1999

[3] Henry McGilton + Mary Campione
PostScript by Example
Addison-Wesley
Reading, Massachusetts ...
1998

[4] Quite Software Ltd
http://www.quite.com

[5] References for Color Science
http://docs-hoffmann.de/ciexyz29082000.pdf

[6] References for PostScript
http://docs-hoffmann.de/pstutor22112002.pdf

[7] Everything about Color and Computers
http://www.efg2.com

[8] G.Hoffmann
Cube Plane Intersections
http://docs-hoffmann.de/cubeplane12112006.pdf
2006

This doc:
http://docs-hoffmann.de/ciegamut16012003.pdf

Gernot Hoffmann
January 16 / 2003 - February 04 / 2013
Website
Load Browser / Click here