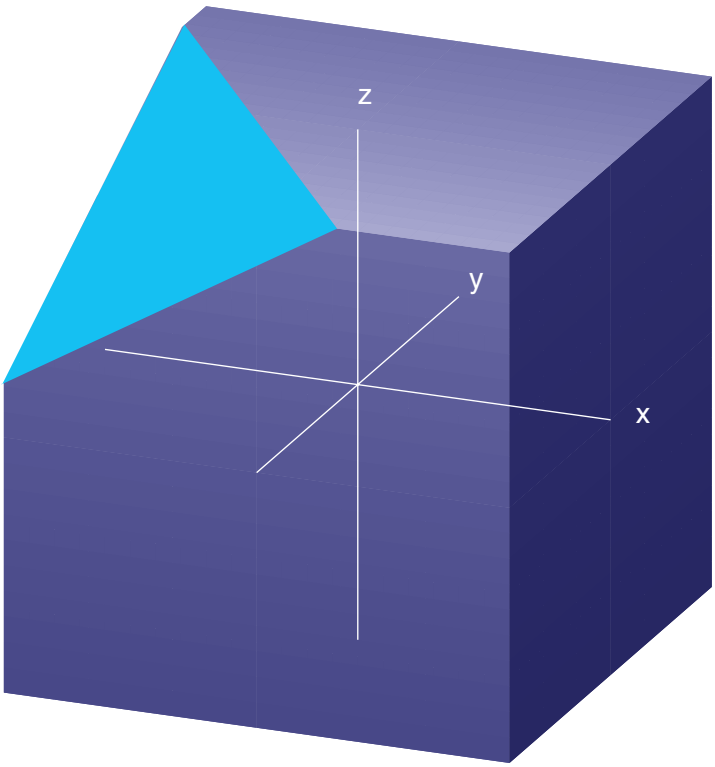
Gernot Hoffmann

Cube Plane Intersection



Contents

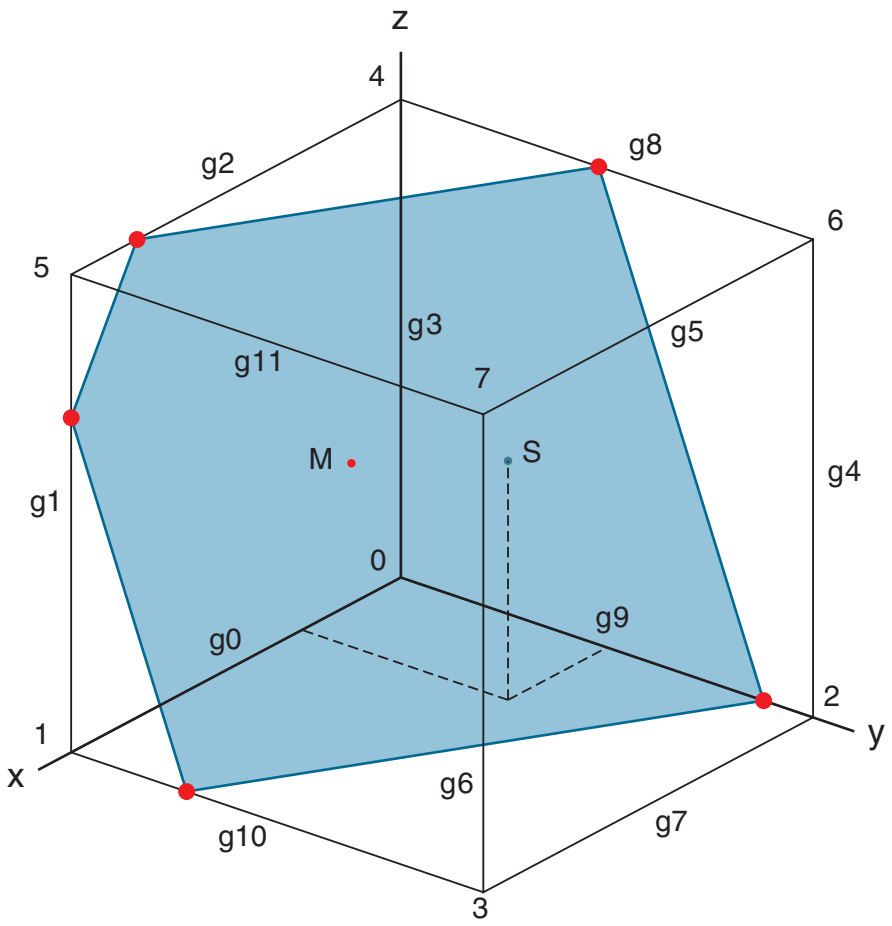
1. Introduction

The task: find the intersection between a cube and an infinite plane. The cube represents as well objects like a square stone (*Quader*) and objects with mutually parallel faces (*Spat)*.

Regular intersections are polygons with 3,4,5 or 6 corners. Irregular intersections are a corner, an edge or a face of the cube.

The unit cube is defined by corners and an edge list. The plane is defined by a point S and a non-zero normal vector **n**.
Intersection points are shown by big red dots. If more than three points were found, then the points have to be sorted so, that the resulting polygon is convex. This is done by calculating the angles in the 'main plane', which is orthogonal to the largest component of **n**. The angle is measured with reference to the mean value M of all intersection points.

A practical application is described in [6], solved by somewhat simplified mathematics (2003).




```
% Corners
/cr [     0 0 0
          0 0 1
          0 1 0
          0 1 1
          1 0 0
          1 0 1
          1 1 0
          1 1 1 ] def

% Edges
/se [     0 1
          1 5
          5 4
          4 0
          2 6
          6 7
          7 3
          3 2
          4 6
          2 0
          1 3
          7 5 ] def
```

2.1 Algorithm / Fundamentals

It is assumed that the reader is familiar with fundamentals of geometry.

Equation of a plane with normal **n** through a point S:

$$\mathbf{n}^T\mathbf{x} = \mathbf{n}^T\mathbf{s}$$

Equation of an edge from Q to R for t=0...1:

$$\mathbf{x} = \mathbf{q} + t(\mathbf{r} - \mathbf{q}) = \mathbf{q} + t\mathbf{g}$$

Distance of a point X_i from the plane, $|\mathbf{n}| = 1$:

$$d_i = \left|\mathbf{n}^T\mathbf{x}_i - \mathbf{n}^T\mathbf{s}\right| = \left|\mathbf{n}^T(\mathbf{x}_i - \mathbf{s})\right|$$

Edge intersection:

$$\mathbf{n}^T(\mathbf{q}_k + t\mathbf{g}_k) = \mathbf{n}^T\mathbf{s}$$

$$t\,\mathbf{n}^T\mathbf{g}_k = \mathbf{n}^T(\mathbf{s} - \mathbf{q}_k)$$

$$t\,g_k = f_k$$

$$t = f_k / g_k$$

Solution if $0 < t < 1$.

2.2 Algorithm / Corner Intersections

The algorithms are explained by pseudocode. It cannot be guaranteed that the pseudocode reflects the real PostScript code accurately. Please refer in any case of doubt to the PS code example.
PS uses index=0 based arrays, the pseudocode uses index=1 based arrays for better understandability.
This program part calculates up to four intersections with corners.

Find 0,1,2 or 4 intersections with 8 corners (i):

```
ε = 10^-8
n_p = 0   % number of intersections
For i=1 To 8 Do
Begin
  d_i = |n^T(x_i - s)|
  If d_i < ε Then
    Begin
      % corner (i) is intersection point
      Inc(n_p)
      Store x_i
      If n_p = 4 Then Exit(for)
    End
End
```

2.3 Algorithm / Edge Intersections

This program part finds altogether up to six edge intersections. If corner intersections were already found, then the total number is limited to six. If an edge intersection is very near to a corner intersection then it will be ignored.

Find intersections with 12 edges (k):
For $k = 1$ To 12 Do
Begin
$g_k = \mathbf{n}^T \mathbf{g}_k$
$f_k = \mathbf{n}^T (\mathbf{s} - \mathbf{q}_k)$
% $t = f_k / g_k$
% Solution if $0 < t < 1$
% Multiply inequality by g_k
fnd = false % fnd = intersection found
If $g_k > 0$ Then
  If $0 < f_k < g_k$ Then fnd = true
If $g_k < 0$ Then
  If $0 > f_k > g_k$ Then fnd = true
If fnd Then
  Begin
    $t = f_k / g_k$
    $\mathbf{x}_k = \mathbf{q}_k + t\mathbf{g}_k$
    skip = false
    For $j = 1$ To n_p Do
    Begin
      $d_k = |x_k - x_j| + |y_k - y_j| + |z_k - z_j|$
      If $d_k < 20\varepsilon$ Then % a somewhat arbitrary threshold
      Begin
        skip = true % ignore double intersection
        Exit(for)
      End
    End
    If Not skip Then
      Begin
        Inc(n_p)
        Store $\mathbf{x}_k$
        If $n_p = 6$ Then Exit(for)
      End
  End
End

2.4 Algorithm / Sorting

For more than three intersection points the points have to be sorted in order to create a convex polygon.
This is done in the ‘main plane’ which is orthogonal to the longest component of the normal vector.

PostScript */angle y x atan def* delivers the angle in the range 0° to 360°. The C/C++ procedure *angle = atan2(y,x)* delivers the angle in the range $-\pi$ to $+\pi$. Add 2π if angle<0.

For polygons with more than three corners it cannot happen that both arguments are zero (in this application).

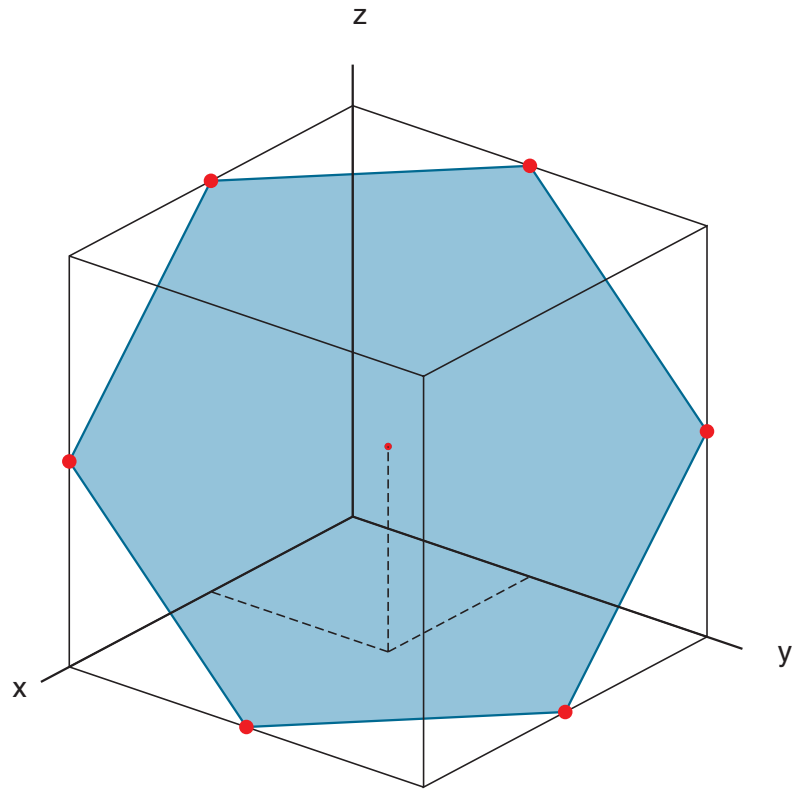
Calculate mean value for intersection points:

$$\mathbf{m} = (1/n_p)\sum_{i=1}^{n_p}\mathbf{x}_i$$

If more than 3 intersections were found then sort.
Find longest component of normal vector and
use the respective ’main plane’ for sorting.
E.g. for $n_z = \max$ use the xy-plane:

For $i = 1$ To $(n_p - 1)$ Do
Begin
$a_i = a\tan2(y_i - m_y, x_i - m_x)$
  Begin
  For $k = i+1$ To n_p Do
    Begin
      $a_k = a\tan2(y_k - m_y, x_k - m_x)$
      If $a_k < a_i$ Then
      Begin
        Swap $\mathbf{x}_i$ and $\mathbf{x}_k$
        $a_i = a_k$
      End
    End
  End
End

3.1 Examples / Regular Cases



Camera pos.+ viewpoint

Px	5.000000
Py	4.000000
Pz	3.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000

Plane point + normal

Sx	0.500000
Sy	0.500000
Sz	0.500000
nx	1.000000
ny	1.000000
nz	1.000000

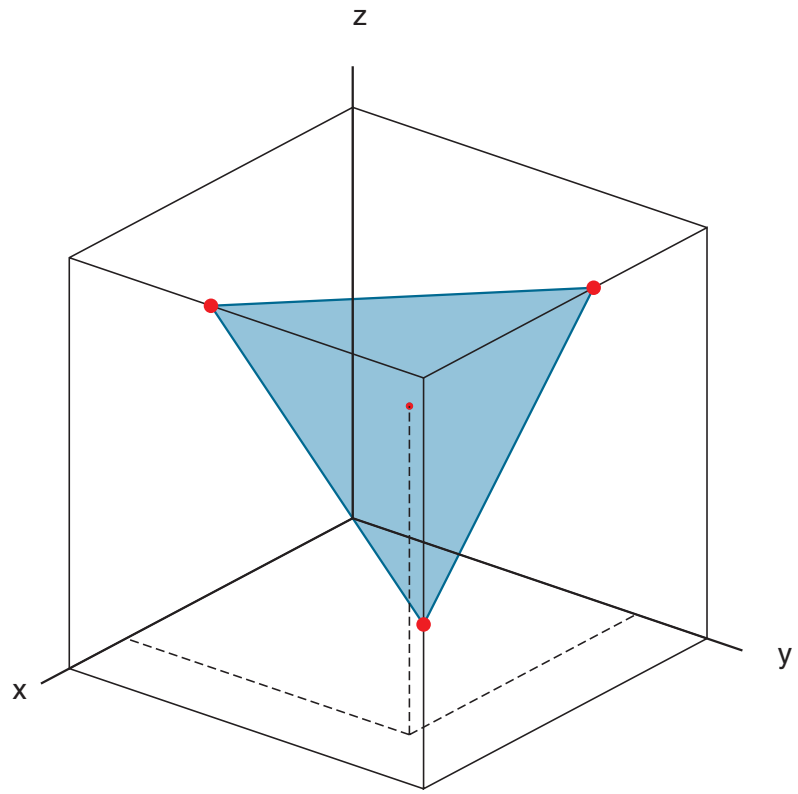
Mean of intersections

Mx	0.500000
My	0.500000
Mz	0.500000

Intersections

Np	6

Example 1 / six edge points



Camera pos.+ viewpoint

Px	5.000000
Py	4.000000
Pz	3.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000

Plane point + normal

Sx	0.800000
Sy	0.800000
Sz	0.800000
nx	1.000000
ny	1.000000
nz	1.000000

Mean of intersections

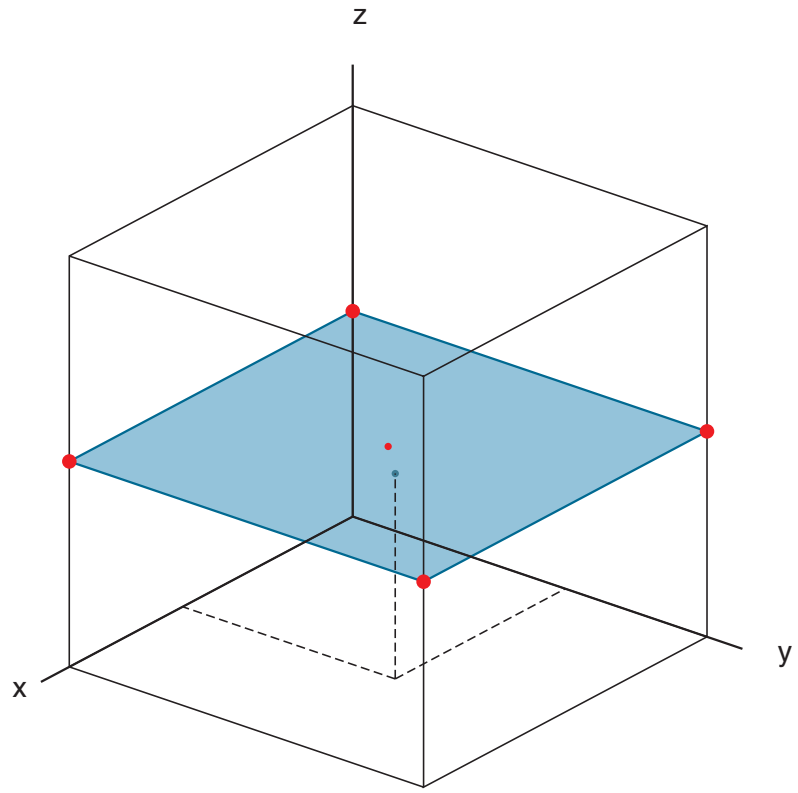
Mx	0.800000
My	0.800000
Mz	0.800000

Intersections

Np	3

Example 2 / three edge points

3.2 Examples / Regular Cases



Example 3 / four edge points

Camera pos.+ viewpoint

Px	5.000000
Py	4.000000
Pz	3.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000

Plane point + normal

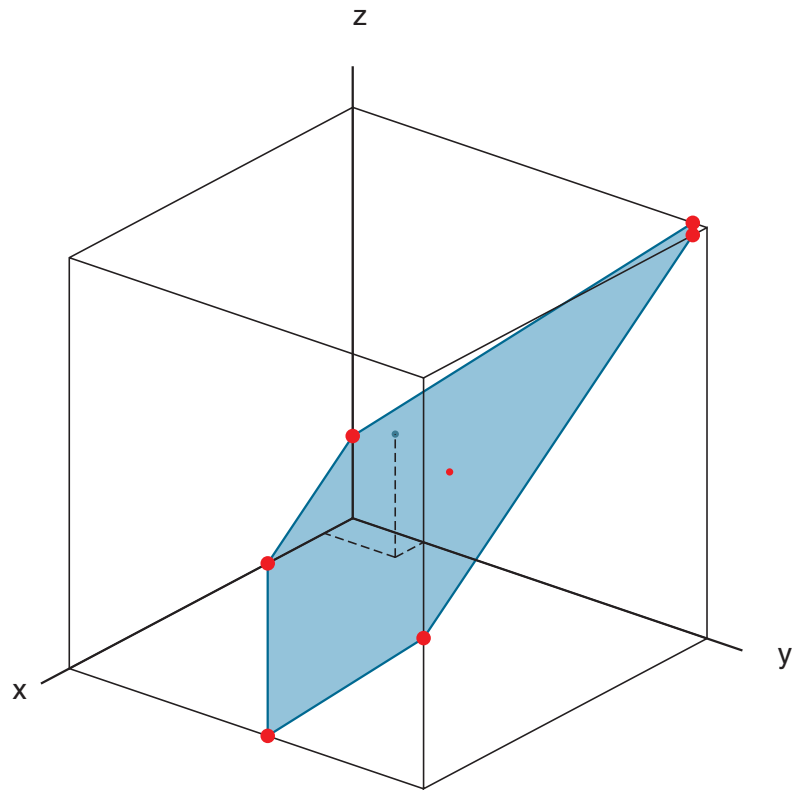
Sx	0.600000
Sy	0.600000
Sz	0.500000
nx	0.000000
ny	0.000000
nz	1.000000

Mean of intersections

Mx	0.500000
My	0.500000
Mz	0.500000

Intersections

Np	4



Example 4 / six edge points

Camera pos.+ viewpoint

Px	5.000000
Py	4.000000
Pz	3.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000

Plane point + normal

Sx	0.100000
Sy	0.200000
Sz	0.300000
nx	0.400000
ny	-0.500000
nz	0.600000

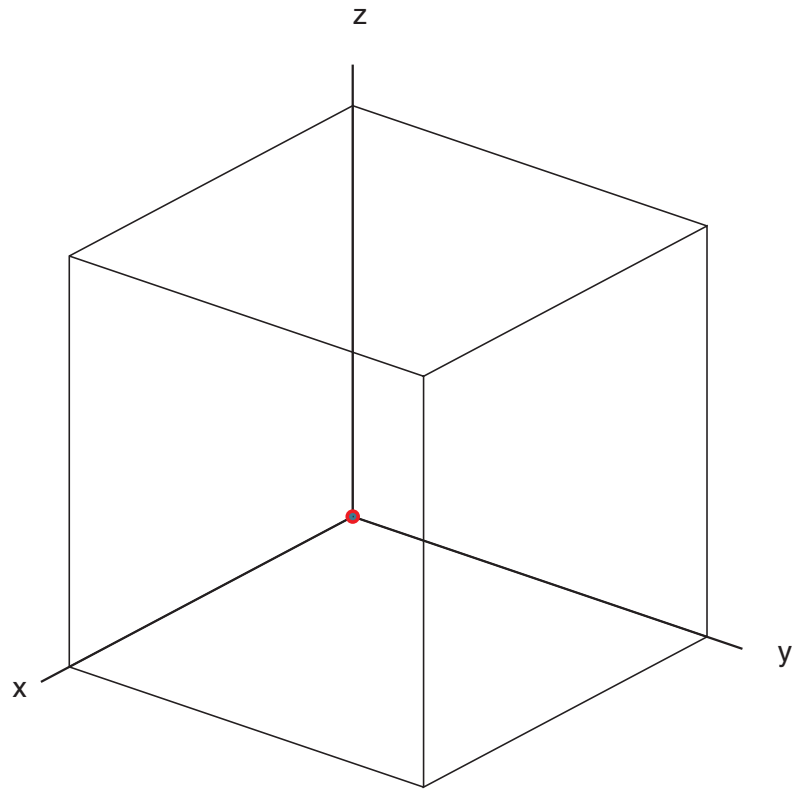
Mean of intersections

Mx	0.391667
My	0.586667
Mz	0.427778

Intersections

Np	6

3.3 Examples / Special Cases



Camera pos.+ viewpoint

Px	5.000000
Py	4.000000
Pz	3.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000

Plane point + normal

Sx	0.000000
Sy	0.000000
Sz	0.000000
nx	1.000000
ny	1.000000
nz	1.000000

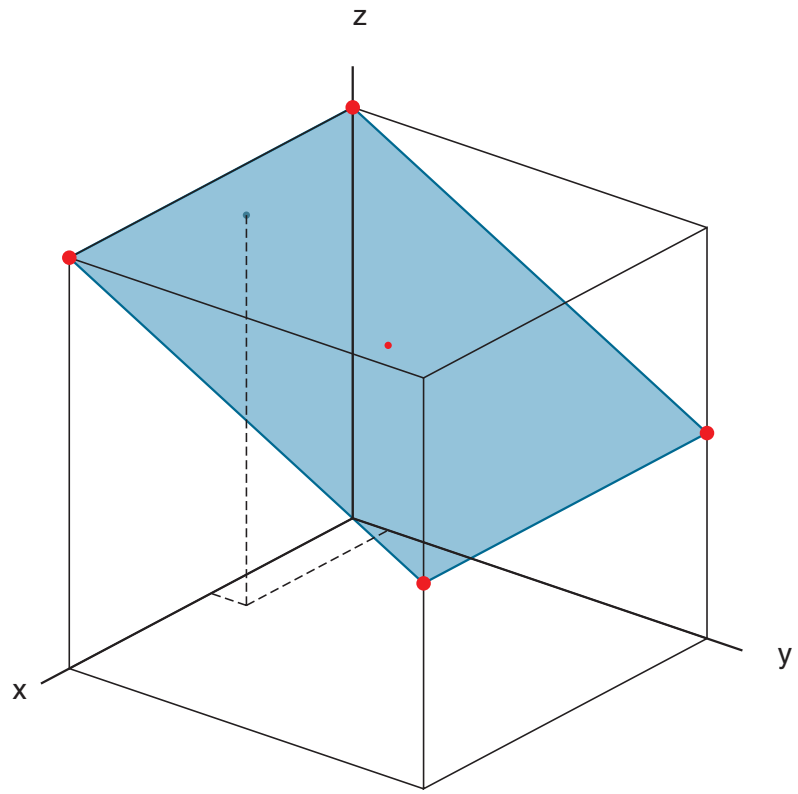
Mean of intersections

Mx	0.000000
My	0.000000
Mz	0.000000

Intersections

Np	1

Example 5 / one corner



Camera pos.+ viewpoint

Px	5.000000
Py	4.000000
Pz	3.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000

Plane point + normal

Sx	0.500000
Sy	0.100000
Sz	0.950000
nx	0.000000
ny	1.000000
nz	2.000000

Mean of intersections

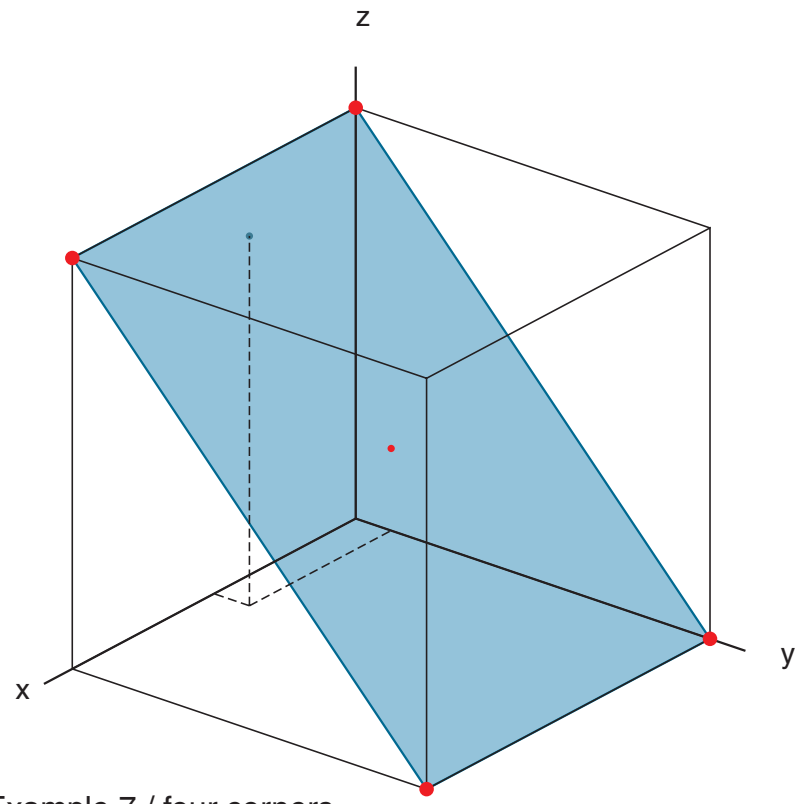
Mx	0.500000
My	0.500000
Mz	0.750000

Intersections

Np	4

Example 6 / two corners, two edge points

3.4 Examples / Special Cases



Example 7 / four corners

Camera pos.+ viewpoint

Px	5.000000
Py	4.000000
Pz	3.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000

Plane point + normal

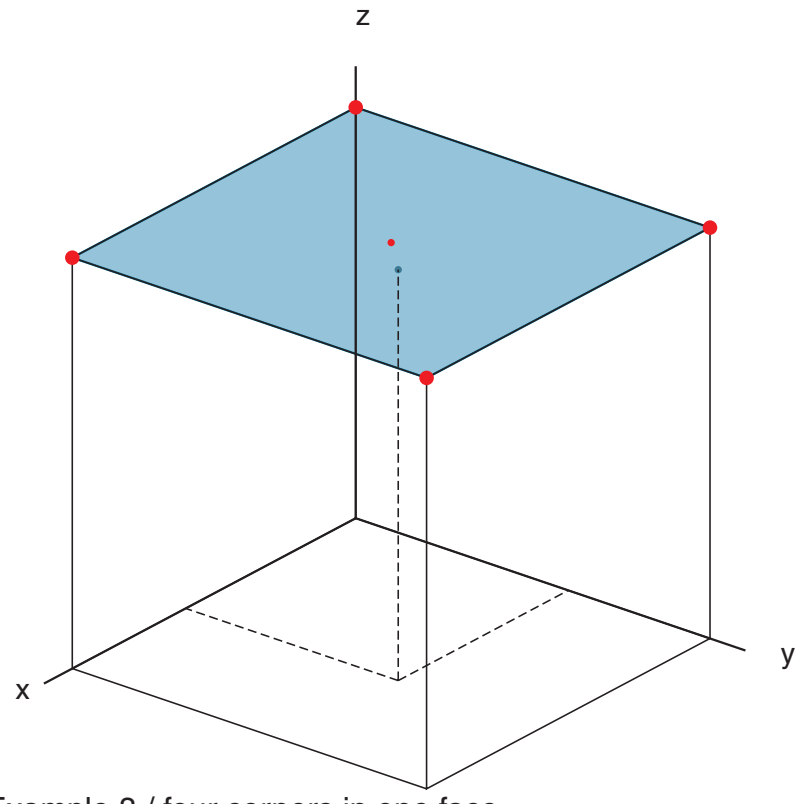
Sx	0.500000
Sy	0.100000
Sz	0.900000
nx	0.000000
ny	1.000000
nz	1.000000

Mean of intersections

Mx	0.500000
My	0.500000
Mz	0.500000

Intersections

Np	4



Example 8 / four corners in one face

Camera pos.+ viewpoint

Px	5.000000
Py	4.000000
Pz	3.000000
Vx	0.000000
Vy	0.000000
Vz	0.000000

Plane point + normal

Sx	0.600000
Sy	0.600000
Sz	1.000000
nx	0.000000
ny	0.000000
nz	1.000000

Mean of intersections

Mx	0.500000
My	0.500000
Mz	1.000000

Intersections

Np	4

4.1 PostScript Code

```
%!PS-Adobe-3.0  EPSF-3.0
%%BoundingBox:  0 0 762 532
%%Creator:      Gernot Hoffmann
%%Title:        CubePlane00
%%CreationDate: November 13/2006

% Disable setpagedevice
 /setpagedevice {pop} bind def

% Intersection of a unit cube and an infinite plane

/Typ 0 def

/mm {2.834646 mul} def

/sc  100 mm def
/bx 760 def
/by 530 def

/lw 0.5 mm sc div def

/x0   90 mm def     % Center
/y0   70 mm def

/MatCam % Camera rotation matrix
{/eps 1E-4 def
/dx Px Vx sub def
/dy Py Vy sub neg def
/dz Pz Vz sub neg def
/dr dx dup mul dy dup mul add sqrt def
  dx abs eps gt dy abs eps gt or
{/Gam dx dy atan def }
{/Gam 0 def          } ifelse
  dz abs eps gt dr abs eps gt or
{/Alf dz dr atan def }
{/Alf 0 def          } ifelse
/ca Alf cos def /sa Alf sin def
/cb Bet cos def /sb Bet sin def
/cg Gam cos def /sg Gam sin def
/c11 cb cg mul sa sb mul sg mul sub def
/c12 cb sg mul sa sb mul cg mul add def
/c13 ca sb mul neg def
/c21 ca sg mul neg def
/c22 ca cg mul def
/c23 sa def
/c31 sb cg mul sa cb mul sg mul add def
/c32 sb sg mul sa cb mul cg mul sub def
/c33 ca cb mul def
/DD  dx dup mul dy dup mul add dz dup mul add sqrt def
} def
```

4.2 PostScript Code

```
/Transf
{% Mod=Cam Camera projection
 %   Map=Par Parallel
 %   Map=Per Perspective
 % xf yf zf on the stack expected
 /zc exch Pz sub def
 /yc exch Py sub def
 /xc exch Px sub def
 /uc c11 xc mul c12 yc mul add c13 zc mul add def
 /wc c31 xc mul c32 yc mul add c33 zc mul add def
  Map (Par) eq
  {/fd uc def /hd wc def } if
  Map (Per) eq
  {/vc c21 xc mul c22 yc mul add c23 zc mul add def
   /fd uc vc div DD mul def /hd wc vc div DD mul def } if
} def

/Box
{ 0 0 0 1 setcmykcolor
  1 setlinewidth
  newpath
  1 1 moveto
  bx 0 rlineto 0 by rlineto bx neg 0 rlineto
  closepath
  stroke
} def

/Axes
{  0 0 0 1 setcmykcolor
   lw setlinewidth
   newpath
   0 0 0 Transf fd hd moveto  1.1 0   0   Transf fd hd lineto
   0 0 0 Transf fd hd moveto  0   1.1 0   Transf fd hd lineto
   0 0 0 Transf fd hd moveto  0   0   1.1 Transf fd hd lineto
   stroke
   0 0 0 1 setcmykcolor
  /fh 18 sc div def
  /Helvetica findfont fh scalefont setfont
   1.2 0   0   Transf fd hd moveto (x) show
   0   1.2 0   Transf fd hd moveto (y) show
   0   0   1.2 Transf fd hd moveto (z) show
 } def

/MinDot
{/hdd exch def  /fdd exch def
 /rdd 0.8 mm sc div def
  newpath fdd hdd rdd 0 360 arc fill
} def

/MaxDot
{/hdd exch def  /fdd exch def
 /rdd 1.6 mm sc div def
  newpath fdd hdd rdd 0 360 arc fill
} def
```

4.3 PostScript Code

```
/Dots
{ 0 1 1 0 setcmykcolor
/kp 0 def
 1 1 np
{ pop
 /xf pt kp       get def
 /yf pt kp 1 add get def
 /zf pt kp 2 add get def
  xf yf zf Transf fd hd MaxDot
 /kp kp 3 add def
} for
 0.5 0 0 0.5 setcmykcolor
 sx sy sz Transf fd hd MinDot
 0 1 1 0 setcmykcolor
 np 3 ge
 {mx my mz Transf fd hd MinDot} if
sx abs 1.5 lt sy abs 1.5 lt and sz abs 1.5 lt and
{ % dashed lines
[0.02 0.01] 0 setdash
lw 0.7 mul setlinewidth
0 0 0 1 setcmykcolor
newpath
sx sy 0 Transf /fs fd def /hs hd def
fs hs moveto
sx 0 0 Transf fd hd lineto
fs hs moveto
0 sy 0 Transf fd hd lineto
fs hs moveto
sx sy sz Transf fd hd lineto
stroke
[] 0 setdash } if
} def

/Dotp
% c=A*B  Matrix [ax ay az]*Matrix [bx by bz]—>c
{/cbz exch def /cby exch def /cbx exch def
 /caz exch def /cay exch def /cax exch def
 /ccd cax cbx mul cay cby mul add caz cbz mul add def
  ccd
} def

/Shownum
% full code in source [5], here  deleted
% Draw number by string
% Version May 15 2004 / uses rounding
{} def

/LF
{/ytxt ytxt fh sub def} def

/List
% full code in source code, here deleted
{} def
```

4.4 PostScript Code

```
% Corners
/cr [ 0 0 0
      0 0 1
      0 1 0
      0 1 1
      1 0 0
      1 0 1
      1 1 0
      1 1 1 ] def

% Edges
/se [ 0 1
      1 5
      5 4
      4 0
      2 6
      6 7
      7 3
      3 2
      4 6
      2 0
      1 3
      7 5 ] def

% Points, up to 6 intersections
/pt [ 0 0 0
      0 0 0
      0 0 0
      0 0 0
      0 0 0
      0 0 0 ] def

/EdgeQRG
{/no exch 2 mul def
 /se1 se no       get def
 /se2 se no 1 add get def
 /cr1 se1 3 mul def
 /cr2 se2 3 mul def
 /qx cr cr1       get def
 /qy cr cr1 1 add get def
 /qz cr cr1 2 add get def
 /rx cr cr2       get def
 /ry cr cr2 1 add get def
 /rz cr cr2 2 add get def
 /gx rx qx sub def
 /gy ry qy sub def
 /gz rz qz sub def
} def
```

4.5 PostScript Code

```
/EdgeDraw
{
0 0 0 1 setcmykcolor
lw 0.7 mul setlinewidth
newpath
0 1 11
{ EdgeQRG % uses no=0..11
  qx qy qz Transf
  fd hd moveto
  rx ry rz Transf
  fd hd lineto
} for
stroke
} def

/FindPts
{
/nxold nx def /nyold ny def /nzold nz def
/nn nx dup mul ny dup mul add nz dup mul add sqrt def
/nx nx nn div def
/ny ny nn div def
/nz nz nn div def

% CheckCorners
/eps1 1e-8 def
/eps2 eps1 20 mul def
/kp 0 def
/kc 0 def
/mx 0 def /my 0 def /mz 0 def
/np 0 def
1 1 8
{ pop
 /xf cr kc       get def
 /yf cr kc 1 add get def
 /zf cr kc 2 add get def
 /dis nx xf sx sub mul ny yf sy sub mul add
      nz zf sz sub mul add abs def
  dis eps1 lt
  {pt kp       xf put
   pt kp 1 add yf put
   pt kp 2 add zf put
  /kp kp 3 add def
  /np np 1 add def
  /mx mx xf add def
  /my my yf add def
  /mz mz zf add def
  } if
   np 4 eq {exit} if
  /kc kc 3 add def
} for
```

4.6 PostScript Code

```
% Check edges
 0 1 11
{ EdgeQRG
 /fx sx qx sub def
 /fy sy qy sub def
 /fz sz qz sub def
 /f nx ny nz fx fy fz Dotp def
 /g nx ny nz gx gy gz Dotp def
 /fnd false def
 g 0 gt
 { f 0 gt f g lt and {/fnd true def } if } if
 g 0 lt
 { f 0 lt f g gt and {/fnd true def } if } if
 fnd
 {/t f g div def
  /xf qx gx t mul add def
  /yf qy gy t mul add def
  /zf qz gz t mul add def
 % Skip double point
  /kj 0 def
  /skip false def
   1 1 np
   {pop
   /xj pt kj       get def
   /yj pt kj 1 add get def
   /zj pt kj 2 add get def
   /dis xf xj sub abs yf yj sub abs add
    zf zj sub abs add def
    dis eps2 lt {/skip true def exit} if
   /kj kj 3 add def
   } for
  skip not
  {
   pt kp       xf put
   pt kp 1 add yf put
   pt kp 2 add zf put
  /kp kp 3 add def
  /np np 1 add def
  /mx mx xf add def
  /my my yf add def
  /mz mz zf add def
  } if
 } if
 np 6 eq {exit} if
 } for
% Mean value
 /mx mx np div def
 /my my np div def
 /mz mz np div def
% Sort by angle
np 3 gt
{/anx nx abs def
 /any ny abs def
 /anz nz abs def
```

```
 1 {
 anx any ge anx anz ge and {PtSortYZ exit} if
 any anx ge any anz ge and {PtSortXZ exit} if
 anz anx ge anz any ge and {PtSortXY     } if
 } repeat
} if
/kp 0 def
 lw setlinewidth
 newpath
 /xf pt kp       get def
 /yf pt kp 1 add get def
 /zf pt kp 2 add get def
  xf yf zf Transf fd hd moveto
 /kp kp 3 add def
 1 1 np 1 sub
 { pop
  /xf pt kp       get def
  /yf pt kp 1 add get def
  /zf pt kp 2 add get def
   xf yf zf Transf fd hd lineto
  /kp kp 3 add def
 } for
 closepath
 np 3 ge
 { gsave
   0.3 0 0 0.15 setcmykcolor
   fill
   grestore
 } if
 1 0 0 0.5 setcmykcolor
 stroke
} def

% Sorting by angles:
% Attention: PostScript versus C/C++
% /ang y x atan def delivers ang=  0..360
%  ang = atan2(y,x) delivers ang=-pi..+pi

/PtSortXY
{/np1 np 1 sub def
 /i3 0 def
 /i1 1 def
 1 1 np1
 {pop
 /xfi pt i3       get def
 /yfi pt i3 1 add get def
 /ai yfi my sub xfi mx sub atan def
 /k3 i3 3 add def
  i1 1 np1
  { pop
   /xfk pt k3       get def
   /yfk pt k3 1 add get def
   /ak yfk my sub xfk mx sub atan def
```

```
   ak ai lt
    {/xbi pt i3       get def
     /ybi pt i3 1 add get def
     /zbi pt i3 2 add get def
      pt i3       pt k3       get put
      pt i3 1 add pt k3 1 add get put
      pt i3 2 add pt k3 2 add get put
     /ai ak def
      pt k3       xbi put
      pt k3 1 add ybi put
      pt k3 2 add zbi put
    } if
   /k3 k3 3 add def
   } for
  /i1 i1 1 add def
  /i3 i3 3 add def
  } for
} def

/PtSortXZ
{/np1 np 1 sub def
 /i3 0 def
 /i1 1 def
 1 1 np1
 {pop
 /xfi pt i3       get def
 /zfi pt i3 2 add get def
 /ai zfi mz sub xfi mx sub atan def
 /k3 i3 3 add def
  i1 1 np1
  { pop
   /xfk pt k3       get def
   /zfk pt k3 2 add get def
   /ak zfk mz sub xfk mx sub atan def
   ak ai lt
    {/xbi pt i3       get def
     /ybi pt i3 1 add get def
     /zbi pt i3 2 add get def
      pt i3       pt k3       get put
      pt i3 1 add pt k3 1 add get put
      pt i3 2 add pt k3 2 add get put
     /ai ak def
      pt k3       xbi put
      pt k3 1 add ybi put
      pt k3 2 add zbi put
    } if
   /k3 k3 3 add def
   } for
  /i1 i1 1 add def
  /i3 i3 3 add def
  } for
} def
```

4.9 PostScript Code

```
/PtSortYZ
{/np1 np 1 sub def
 /i3 0 def
 /i1 1 def
 1 1 np1
 {pop
 /yfi pt i3 1 add get def
 /zfi pt i3 2 add get def
 /ai zfi mz sub yfi my sub atan def
 /k3 i3 3 add def
  i1 1 np1
  { pop
   /yfk pt k3 1 add get def
   /zfk pt k3 2 add get def
   /ak zfk mz sub yfk my sub atan def
   ak ai lt
    {/xbi pt i3       get def
     /ybi pt i3 1 add get def
     /zbi pt i3 2 add get def
      pt i3       pt k3       get put
      pt i3 1 add pt k3 1 add get put
      pt i3 2 add pt k3 2 add get put
     /ai ak def
      pt k3       xbi put
      pt k3 1 add ybi put
      pt k3 2 add zbi put
    } if
   /k3 k3 3 add def
   } for
  /i1 i1 1 add def
  /i3 i3 3 add def
  } for
} def

true setstrokeadjust % for monitor

gsave
Box

x0 y0 translate
sc sc scale

/Px   5   def  % Camera position
/Py   4   def
/Pz   3   def
/Vx   0   def  % View point
/Vy   0   def
/Vz   0   def
/Bet  0   def  % Roll angle
```

4.10 PostScript Code

```
% /Typ 0 def % defined in header
Typ 0 eq % General illustration, pentagon
{/sx  0.3  def  % Point on plane
 /sy  0.5  def
 /sz  0.5  def
 /nx  0.6  def  % Plane normal
 /ny  1.0  def
 /nz  0.4  def  } if
Typ 1 eq % Hexagon
{/sx  0.5  def  % Point on plane
 /sy  0.5  def
 /sz  0.5  def
 /nx  1.0  def  % Plane normal
 /ny  1.0  def
 /nz  1.0  def  } if
Typ 2 eq % Triangle
{/sx  0.8  def  % Point on plane
 /sy  0.8  def
 /sz  0.8  def
 /nx  1.0  def  % Plane normal
 /ny  1.0  def
 /nz  1.0  def  } if
Typ 3 eq % Square
{/sx  0.6  def  % Point on plane
 /sy  0.6  def
 /sz  0.5  def
 /nx  0.0  def  % Plane normal
 /ny  0.0  def
 /nz  1.0  def  } if
Typ 4 eq % Random
{/sx  0.1  def  % Point on plane
 /sy  0.2  def
 /sz  0.3  def
 /nx  0.4  def  % Plane normal
 /ny -0.5  def
 /nz  0.6  def  } if
Typ 5 eq % Single corner
{/sx  0.0  def  % Point on plane
 /sy  0.0  def
 /sz  0.0  def
 /nx  1.0  def  % Plane normal
 /ny  1.0  def
 /nz  1.0  def  } if
Typ 6 eq % Two corners
{/sx  0.5  def  % Point on plane
 /sy  0.1  def
 /sz  0.95 def
 /nx  0.0  def  % Plane normal
 /ny  1.0  def
 /nz  2.0  def  } if
```

4.11 PostScript Code

```
Typ 7 eq % Four corners
{/sx  0.5  def  % Point on plane
 /sy  0.1  def
 /sz  0.9  def
 /nx  0.0  def  % Plane normal
 /ny  1.0  def
 /nz  1.0  def  } if
Typ 8 eq % Face
{/sx  0.6  def  % Point on plane
 /sy  0.6  def
 /sz  1.0  def
 /nx  0.0  def  % Plane normal
 /ny  0.0  def
 /nz  1.0  def  } if

/Map (Par) def % Parallel perspective
MatCam
FindPts
Axes
EdgeDraw
Dots
grestore
List

showpage
```

5. References

[1] PostScript Language Reference (PS3)
Addison-Wesley, Boston, San Francisco ...
2002

[2] H.McGilton + M.Campione
PostScript by Example
Addison-Wesley, Reading, Massachusetts ...
1998

[3] J.D.Foley et al.
Computer Graphics
Addison-Wesley, Reading, Massachusetts ...
1990

[4] G.Hoffmann
Planar Projections
http://docs-hoffmann.de/project18032004.pdf

[5] G.Hoffmann
Source code for Cube Plane Intersection
Replace *.txt by *.eps
http://docs-hoffmann.de/cubeplane00.txt

[6] G.Hoffmann
Gamut for CIE and sRGB Primaries with Varying Luminance
http://docs-hoffmann.de/ciegamut16012003.pdf
2003

This document

http://docs-hoffmann.de/cubeplane12112006.pdf

Gernot Hoffmann
November 12 / 2006 — February 14 / 2013
Website
Load browser / Click here