Gernot Hoffmann

Distance between Line Segments

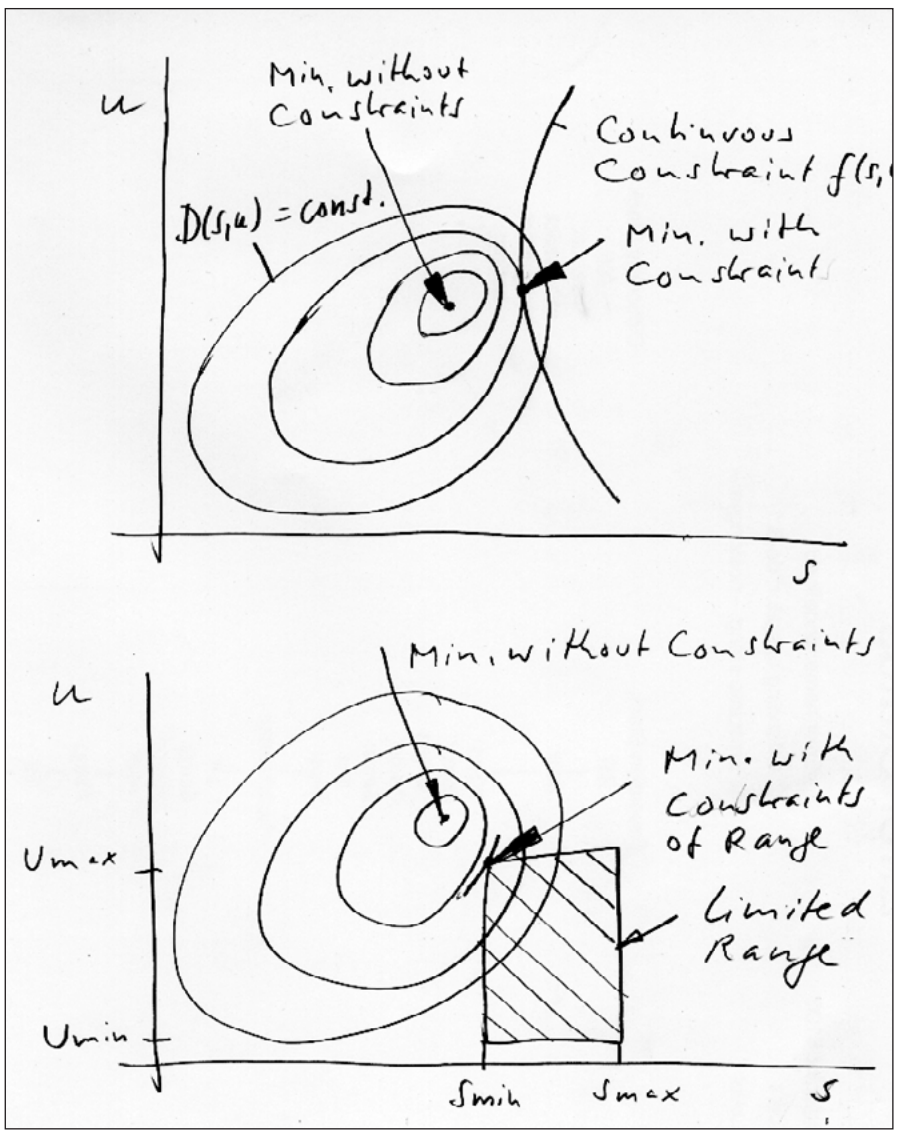


Table of Contents

1. Introduction

The (shortest) distance between two *infinite lines* is found by minimizing the squared distance between two arbitrary points $\mathbf{x}_1$ and $\mathbf{x}_2$. Result is a system of two linear equations for the two line parameters s_1,s_2 in the functions $\mathbf{x} = \mathbf{p} + s\mathbf{g}$.
This fails for the case of parallel lines and identical lines because the solution is not unique. The case is easily analyzed and therefore not a practical problem.
One should not base the calculation on the assumption that the shortest connection between the lines is orthogonal to both lines. This will fail for the trivial situation that the lines have a true intersection and therefore a zero length connection vector.

For the (shortest) distance of two *line segments* the solution is not as simple. The connection vector is either 'in-line' for both segments (the end points are on the segment) or in-line for one segment or not in-line for both. In these cases the solution is found by searching the shortes distances for all four end points of the line segments. It is either a straight connection between end points or a line from one end point orthogonal to the other line.

The suggested algorithm is very robust. It works as well for line segments which are degenerated to points. Even for the distance of two identical points which is of course zero.
In case of doubts please refer to the Pascal source code.

The title illustration shows the minimum search for constrained and for bounded parameters. For bounded parameters the minimum is either a relative minimum inside the bounding area or an absolute minimum on the boundary.

Settings for Acrobat (relevant only for page 5)

Edit / Preferences / General / Page Display (since version 6)
Custom Resolution 72 dpi

Edit / Preferences / General / Color Management (full version only)
sRGB
EuroscaleCoated or ISOCoated or SWOP
GrayGamma 2.2

2.1 Algorithm

Given are two line segments from $\mathbf{p}_i$ to $\mathbf{q}_i$

Direction

$\mathbf{g}_1 = \mathbf{q}_1 - \mathbf{p}_1$

$\mathbf{g}_2 = \mathbf{q}_2 - \mathbf{p}_2$

Two arbitrary points, parameters s_1, s_2

$\mathbf{x}_1 = \mathbf{p}_1 + s_1 \mathbf{g}_1$

$\mathbf{x}_2 = \mathbf{p}_2 + s_2 \mathbf{g}_2$

Distance vector

$\mathbf{d} = \mathbf{x}_2 - \mathbf{x}_1 = \mathbf{r} + s_2 \mathbf{g}_2 - s_1 \mathbf{g}_1$

$\mathbf{r} = \mathbf{p}_2 - \mathbf{p}_1$ (in the code $\mathbf{r}$ is named dp)

Squared distance

$F(s_1, s_2) = \mathbf{d}^T \mathbf{d}$

$F = \mathbf{r}^T \mathbf{r} - 2 s_1 \mathbf{g}_1^T \mathbf{r} + 2 s_2 \mathbf{g}_2^T \mathbf{r} + s_1^2 \mathbf{g}_1^T \mathbf{g}_1 + s_2^2 \mathbf{g}_2^T \mathbf{g}_2 - 2 s_1 s_2 \mathbf{g}_1^T \mathbf{g}_2$

Minimize F

$$\frac{\partial F}{\partial s_1} = 0$$

$$\frac{\partial F}{\partial s_2} = 0$$

Results in two linear equations (symmetric)

$a_{11} s_1 + a_{12} s_2 = b_1$

$a_{12} s_1 + a_{22} s_2 = b_2$

Coefficients

$a_{11} = \mathbf{g}_1^T \mathbf{g}_1$

$a_{22} = \mathbf{g}_2^T \mathbf{g}_2$

$a_{12} = -\mathbf{g}_1^T \mathbf{g}_2$

$b1 = \mathbf{g}_1^T \mathbf{r}$

$b2 = -\mathbf{g}_2^T \mathbf{r}$

Cramer determinants

$D_0 = a_{11} a_{22} - a_{12}^2$

$D_1 = b_1 a_{22} - b_2 a_{12}$

$D_2 = b_2 a_{11} - b_1 a_{12}$

Check

$0 < s_1 < 1$

$0 < s_2 < 1$

If $D_0 > 0$

$0 < D_1 < D_0$

$0 < D_2 < D_0$

If $D_0 < 0$

$0 > D_1 > D_0$

$0 > D_2 > D_0$

If s_1 and s_2 are in limits then calculate

$s_1 = D_1 / D_0$

$s_2 = D_2 / D_0$

Otherwise check end points

2.2 Algorithm

Check end points

$s_1 = \text{const}$

$s_2 = \dfrac{b_2 - a_{12}s_1}{a_{22}} = \dfrac{n_2}{a_{22}}$

$s1 = 0$

Check

$0 < s_2 < 1$

If $n_2 \le 0$ then $s_2 = 0$

If $n_2 \ge a_{22}$ then $s_2 = 1$

If s_2 is in limits then calculate $s_2 = n_2 / a_{22}$

Calculate distance d_{11}

$s1 = 1$

Check

$0 < s_2 < 1$

If $n_2 \le 0$ then $s_2 = 0$

If $n_2 \ge a_{22}$ then $s_2 = 1$

If s_2 is in limits then calculate $s_2 = n_2 / a_{22}$

Calculate distance d_{12}

$s_2 = \text{const}$

$s_1 = \dfrac{b_1 - a_{12}s_2}{a_{11}} = \dfrac{n_1}{a_{11}}$

$s_2 = 0$

Check

$0 < s_1 < 1$

If $n_1 \le 0$ then $s_1 = 0$

If $n_1 \ge a_{11}$ then $s_1 = 1$

If s_1 is in limits then calculate $s_1 = n_1 / a_{11}$

Calculate distance d_{21}

$s2 = 1$

Check

$0 < s_1 < 1$

If $n_1 \le 0$ then $s_1 = 0$

If $n_1 \ge a_{11}$ then $s_1 = 1$

If s_1 is in limits then calculate $s_1 = n_1 / a_{11}$

Calculate distance d_{22}

Use case with smallest distance $d_{11}, d_{12}, d_{21}, d_{22}$
and the appropriate set s_1, s_2 (refer to source code)

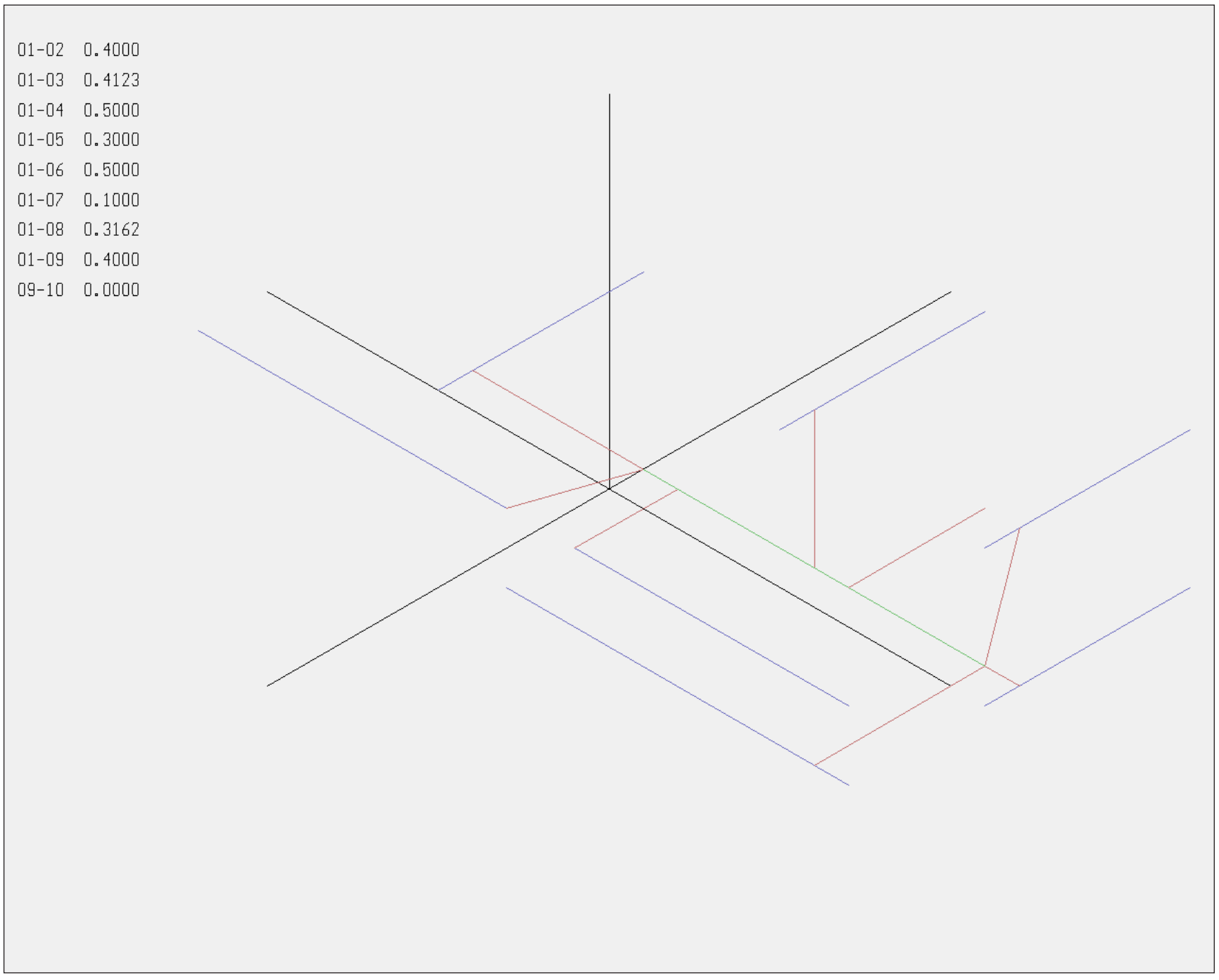
3. Example

The example shows the line segment 1 (green), the line segments 2 to 8 (blue) and the nearest connections (red).

Line segment 9 is a point. This is connected with line segment 1 correctly.

Line segment 10 is the same point as 9. The connection has zero length.

Best view 72dpi / zoom 200%



4.1 Code

This Pascal program uses external graphics libraries.The line segment algorithm is complete.

```
Program ZSegDist;
{ Project  Distance of Line Segments
  Author   G.Hoffmann
  Date     July 03, 2004  }

{$A+,B-,D-,E-,F-,G+,I+,L+,N+,O-,P-,Q-,R-,S-,T-,V-,X-,Y-}
{$C Moveable PreLoad Permanent }
{$M,65500} { Select Protected Mode ! Stack Size        }
Uses Crt,WinDos,
     Zefir30,Zefir31,Zefir32,Zefir33,Zefir34,Zefir35,
     Zefir36,Zefir37,Zefir38,Zefir39,Zefir40;

Var  p1,q1,p2,q2  : XYZ;
     i,k1,k2,sel  : Integer;
     dis          : Double;
Var  Im0,txt1,txt2: String;
Const Lin:  Array [1..10,1..6] of Single =
      (( 0.0, 0.1, 0.0,  1.0, 0.1, 0.0),   { 01 }
       ( 0.5, 0.0, 0.4,  0.5, 0.6, 0.4),   { 02 }
       ( 1.1, 0.0, 0.4,  1.1, 0.6, 0.4),   { 03 }
       (-0.5, 0.0, 0.0, -0.5, 0.6, 0.0),   { 04 }
       ( 0.1,-0.2, 0.0,  0.9,-0.2, 0.0),   { 05 }
       ( 0.1,-0.4, 0.0,  1.1,-0.4, 0.0),   { 06 }
       ( 1.1, 0.0, 0.0,  1.1, 0.6, 0.0),   { 07 }
       (-1.0,-0.2, 0.0, -0.1,-0.2, 0.0),   { 08 }
       ( 0.6, 0.5, 0.0,  0.6, 0.5, 0.0),   { 09 }
       ( 0.6, 0.5, 0.0,  0.6, 0.5, 0.0));  { 10 }
{Pini}
Procedure DifV (P,Q: XYZ; Var R: XYZ);                          Forward;
Procedure DotV (P,Q: XYZ; Var d: Double);                       Forward;
Procedure VecX (P,G: XYZ; s: Double; Var X: XYZ);               Forward;
Procedure DisA (X1,X2: XYZ; Var Da: XYZ; Var Lq: Double);       Forward;
Procedure Dist (p1,q1,p2,q2: XYZ; Var x1,x2: XYZ;
                Var dis: Double);                               Forward;
Procedure Segm (k1,k2: Integer; Var p1,q1,p2,q2: XYZ);          Forward;
Procedure Draw (p1,q1,p2,q2,x1,x2: XYZ; sel: Integer);          Forward;
Procedure Axes (sel: Integer);                                  Forward;
{Pend}

Procedure DifV (P,Q: XYZ; Var R: XYZ);
Begin
 With R Do
 Begin
 x:=P.x-Q.x; y:=P.y-Q.y; z:=P.z-Q.z;
 End;
End;

Procedure DotV (P,Q: XYZ; Var d: Double);
Begin
 d:=P.x*Q.x+P.y*Q.y+P.z*Q.z;
End;

Procedure VecX (P,G: XYZ; s: Double; Var X: XYZ);
Begin
 With X do
 Begin
 x:=P.x+s*G.x; y:=P.y+s*G.y; z:=P.z+s*G.z;
 End;
End;

Procedure DisA (X1,X2: XYZ; Var Da: XYZ; Var Lq: Double);
Begin
 With Da do
 Begin
 x:=X2.x-X1.x; y:=X2.y-X1.y; z:=X2.z-X1.z;
 Lq:=Sqr(x)+Sqr(y)+Sqr(z);
 End;
End;
```

4.2 Code

```
Procedure Dist(p1,q1,p2,q2: XYZ; Var x1,x2: XYZ; Var dis: Double);
Var  a11,a12,a22,b1,b2,D0,D1,D2    : Double;
     s1,s2,n1,n2,sa,sb,dq,eq       : Double;
     g1,g2,dp,da                   : XYZ;
     flg                           : Integer;
Begin
DifV(q1,p1,g1);
DifV(q2,p2,g2);
DifV(p2,p1,dp);
DotV(g1,g1,a11);
DotV(g1,g2,a12); a12:=-a12;
DotV(g2,g2,a22);
DotV(g1,dp,b1);
DotV(g2,dp,b2);  b2:=-b2;
{ Cramer determinants }
D0:=a11*a22-a12*a12;
D1:= b1*a22- b2*a12;
D2:=-b1*a12+ b2*a11;
{ In-line distance }
flg:=0;
If D0>0 Then
Begin
 If (0<D1) And (D1<D0) Then Inc(flg);
 If (0<D2) And (D2<D0) Then Inc(flg);
End;
If D0<0 Then
Begin
 If (0>D1) And (D1>D0) Then Inc(flg);
 If (0>D2) And (D2>D0) Then Inc(flg);
End;
If flg=2 Then
Begin
 sa:=D1/D0; sb:=D2/D0;
End;
{ Out-line distance }
If flg<2 Then
Begin
{ a11>=0, a22>=0 }
 s1:=0;
 n2:=b2;          { s2:=(b2-a12*s1)/a22 }
 flg:=0;
 If n2<=0      Then Begin s2:=0; Inc(flg); End;
 If n2>=a22    Then Begin s2:=1; Inc(flg); End;
 If flg=0  Then s2:=n2/a22;
 VecX (p1,g1,s1,x1);
 VecX (p2,g2,s2,x2);
 DisA (x1,x2,da,eq);
sa:=s1; sb:=s2; dq:=eq;
s1:=1;
n2:=b2-a12;     { s2:=(b2-a12*s1)/a22 }
flg:=0;
If n2<=0      Then Begin s2:=0; Inc(flg); End;
If n2>=a22    Then Begin s2:=1; Inc(flg); End;
If flg=0  Then s2:=n2/a22;
VecX (p1,g1,s1,x1);
VecX (p2,g2,s2,x2);
DisA (x1,x2,da,eq);
If eq<dq Then
 Begin sa:=s1; sb:=s2; dq:=eq; End;
s2:=0;
n1:=b1;          { s1=(b1-a12*s2)/a11 }
flg:=0;
If n1<=0      Then Begin s1:=0; Inc(flg); End;
If n1>=a11    Then Begin s1:=1; Inc(flg); End;
If flg=0  Then s1:=n1/a11;
VecX (p1,g1,s1,x1);
VecX (p2,g2,s2,x2);
DisA (x1,x2,da,eq);
If eq<dq Then
 Begin sa:=s1; sb:=s2; dq:=eq; End;
```

4.3 Code

```
 s2:=1;
 n1:=b1-a12;     { s1=(b1-a12*s2)/a11 }
 flg:=0;
 If n1<=0      Then Begin s1:=0; Inc(flg); End;
 If n1>=a11    Then Begin s1:=1; Inc(flg); End;
 If flg=0  Then s1:=n1/a11;
 VecX (p1,g1,s1,x1);
 VecX (p2,g2,s2,x2);
 DisA (x1,x2,da,eq);
 If eq<dq Then
  Begin sa:=s1; sb:=s2; dq:=eq; End;
 End;
{ For all sa=s1, sb=s2, refresh }
 VecX (p1,g1,sa,x1);
 VecX (p2,g2,sb,x2);
 DisA (x1,x2,da,dq);
 dis:=Sqrt(dq);
End;

Procedure Segm(k1,k2: Integer; Var p1,q1,p2,q2: XYZ);
Begin
With p1 Do
 Begin x:=Lin[k1,1]; y:=Lin[k1,2]; z:=Lin[k1,3];
 End;
With q1 Do
 Begin x:=Lin[k1,4]; y:=Lin[k1,5]; z:=Lin[k1,6];
 End;
With p2 Do
 Begin x:=Lin[k2,1]; y:=Lin[k2,2]; z:=Lin[k2,3];
 End;
With q2 Do
 Begin x:=Lin[k2,4]; y:=Lin[k2,5]; z:=Lin[k2,6];
 End;
End;

Procedure Draw (p1,q1,p2,q2,x1,x2: XYZ; sel: Integer);
Var pp1,pq1,pp2,pq2,px1,px2   : PQE;
    c1,c2,cx                  : PC;
Begin
c1.p:= 60; c1.c:=150;
c2.p:=120; c2.c:=150;
cx.p:=  0; cx.c:=150;
Abbild3R (p1,pp1,sel); { 3D float t0 2D Raster }
Abbild3R (q1,pq1,sel);
Abbild3R (p2,pp2,sel);
Abbild3R (q2,pq2,sel);
Abbild3R (x1,px1,sel);
Abbild3R (x2,px2,sel);
DrawSLine(pp1,pq1,c1,c1,sel);
DrawSLine(pp2,pq2,c2,c2,sel);
DrawSLine(px1,px2,cx,cx,sel);
End;

Procedure Axes(sel: Integer);
Var  pp1,pp2                  : PQE;
     px1,py1,pz1,px2,py2,pz2  : XYZ;
     c1                       : PC;
Begin
c1.p:=181; c1.c:=0;
With px1 Do Begin x:=-1; y:= 0; z:= 0; End;
With py1 Do Begin x:= 0; y:=-1; z:= 0; End;
With pz1 Do Begin x:= 0; y:= 0; z:= 0; End;
With px2 Do Begin x:= 1; y:= 0; z:= 0; End;
With py2 Do Begin x:= 0; y:= 1; z:= 0; End;
With pz2 Do Begin x:= 0; y:= 0; z:= 1; End;
Abbild3R (px1,pp1,sel); { 3D float to 2D Raster }
Abbild3R (px2,pp2,sel);
```

4.4 Code

```
DrawSLine(pp1,pp2,c1,c1,sel);
Abbild3R (py1,pp1,sel);
Abbild3R (py2,pp2,sel);
DrawSLine(pp1,pp2,c1,c1,sel);
Abbild3R (pz1,pp1,sel);
Abbild3R (pz2,pp2,sel);
DrawSLine(pp1,pp2,c1,c1,sel);
End;

BEGIN
Configure
(ConfMax,ZebraMax,Flag,Conf,ZebrArry,
 ZebrPath,SmdrPath,
 BuffDrNm,SnapDrNm,ZebrNm,
 LastDriv,SourDriv,DestDriv,
 MonFrequ,GcardIdy,VesaMode,VesaCode,
 SoundsOn,NoClkInt,AutoCatG,ShowIcon);
SnapPtr:=@SnapShot;
SmartText;
VesaStart(VesaMode);
MemGStart;
Coords(1,0);
camox:= 1 ; camoy:=-1; camoz:= 1;  { Camera position        }
viewx:= 0 ; viewy:= 0; viewz:= 0;  { Camera viewpoint       }
vcam :=1;                          { Vertical Rectification  }
vcam :=0;                          { Perspective             }
vcam :=2;                          { Parallel                }
zcam:=0.8;
MatCam3D;
ColToScr(181,240);
Axes(0);
{ Line-Line
  Line-Point }
k1:=1;
For k2:=2 to 9 do
Begin
Segm (k1,k2,p1,q1,p2,q2);
Dist (p1,q1,p2,q2,x1,x2,dis);
Draw (p1,q1,p2,q2,x1,x2,0);
Str(k2:1,txt1); txt1:='01-0'+txt1;
Str(dis:8:4,txt2); txt1:=txt1+txt2;
WrTxtWXY(1,blac,2,k2,txt1);
End;
{ Point-Point }
k1:= 9;
k2:=10;
Segm (k1,k2,p1,q1,p2,q2);
Dist (p1,q1,p2,q2,x1,x2,dis);
Draw (p1,q1,p2,q2,x1,x2,0);
Str(k2:1,txt1); txt1:='09-'+txt1;
Str(dis:8:4,txt2); txt1:=txt1+txt2;
WrTxtWXY(1,blac,2,k2,txt1);
Im0:='J:\SegDist\SegDist100.bmp';
SaveImag(Im0);
Stop;
MemGEnde;
VesaEnde;
END.
```

5. References

No special references

This doc:

http://docs-hoffmann.de/xsegdist03072004.pdf

Gernot Hoffmann
July 03 / 2004 — February 19 / 2013
Website
Load Browser / Click here