1 Distance between lines, segments and spherical caps

1.1 Distance between lines

First, consider lines

$$\begin{array}{rcl} P(s) & = & P_0 + s\,\mathbf{u} \\ Q(t) & = & Q_o + t\,\mathbf{v} \end{array}$$

a segment between the two lines is

$$\mathbf{w}(s,t) = P(s) - Q(t)$$

the minimum distance segment $\mathbf{w}^c = \mathbf{w}(s_c, t_c)$ is $\perp$ to the lines:

$$\begin{array}{rcl} \mathbf{w}^c \cdot \mathbf{u} & = & 0 \\ -\mathbf{w}^c \cdot \boldsymbol{v} & = & 0 \end{array}$$

i.e. (indicating with $a = \mathbf{u} \cdot \mathbf{u}$, $b = \mathbf{u} \cdot \mathbf{v}$, $c = \mathbf{v} \cdot \mathbf{v}$, $\mathbf{w}^0 = \mathbf{w}(0,0) = P_0 - Q_0$, $d = \mathbf{u} \cdot \mathbf{w}^0$, $e = \mathbf{v} \cdot \mathbf{w}^0$ and $\Delta = ac - b^2$)

$$\begin{array}{rcl} d + s_c\,a - t_c\,b & = & 0 \\ -e - s_c\,b + t_c\,c & = & 0 \end{array}$$

that is

$$\begin{array}{rcl} s_c & = & (be - cd)/\Delta \\ t_c & = & (ae - bd)/\Delta \end{array}$$

Note that $\Delta = \mathbf{u}^2\mathbf{v}^2 - |\mathbf{u}|\,|\mathbf{v}|\cos^2\theta = (|\mathbf{u}|\,|\mathbf{v}|\sin\theta)^2 \geq 0$ and $\Delta = 0$ only when $\mathbf{u} \parallel \mathbf{v}$. In the $\Delta = 0$ case, we can use $s_c = 0$ so that $t_c = e/c$.

2 Distance between segments

We describe finite segments limiting $-1 \leq s \leq 1$; this way $P(s)$ describes a segment centered in P_0, of direction $\mathbf{u}$ and length $2\,|\mathbf{u}|$.

2.1 Distance between spherical caps

xxx

Algorithm 1 Distance between two segments.

```
Vector QP = P-Q;
double a = u*u; // always > 0
double b = u*v;
double c = v*v; // always > 0
double d = u*QP;
double e = v*QP;
double eps = 1.0e-6;
double D = a*c - b*b;     // always >= 0
if (D < eps) { // segment almost parallel
   // decide wich edge of P(s) is nearest
   if( d<0 ) sc = -1.0;
   else sc = 1.0;
   // minimize distance to Q(t)
   tc = (e+sc*b)/c;
}
else{ // closest points on the lines
     sc = (b*e-c*d) / D;
     // check sc is on P(s) and
     // minimize distance
     if(sc<-1.0 )
       { sc = -1.0; tc = (e+sc*b)/c;}
     else if( sc>1.0 )
       { sc = 1.0; tc = (e+sc*b)/c;}
     else tc = (a*e-b*d) / D;
    }

// check tc is on Q(t) and
// minimize distance
if(tc<-1.0) { tc=-1.0; sc=(tc*b-d)/a;
else if(tc>1.0) { tc=1.0; sc=(tc*b-d)/a;}
// last check sc is on P(s)
if(sc<-1.0) sc=-1.0; else if(sc>1.0) sc=1.0;
```