# Distance Between Two Line Segments in 3D

David Eberly
Geometric Tools, LLC
http://www.geometrictools.com/

Created: March 2, 1999
Last Modified: March 1, 2008

## Contents

# 1 Mathematical Formulation

The problem is to compute the minimum distance between points on two line segments $\mathbf{L}_0(s) = \mathbf{B}_0 + s\mathbf{M}_0$ for $s \in [0,1]$, and $\mathbf{L}_1(t) = \mathbf{B}_1 + t\mathbf{M}_1$ for $t \in [0,1]$. The minimum distance is computed by locating the values $\bar{s} \in [0,1]$ and $\bar{t} \in [0,1]$ corresponding to the two closest points on the line segments.

The squared-distance function for any two points on the line segments is $Q(s,t) = |\mathbf{L}_0(s) - \mathbf{L}_1(t)|^2$ for $(s,t) \in [0,1]^2$. The function is quadratic in $s$ and $t$,

$$Q(s,t) = as^2 + 2bst + ct^2 + 2ds + 2et + f,$$

where $a = \mathbf{M}_0 \cdot \mathbf{M}_0$, $b = -\mathbf{M}_0 \cdot \mathbf{M}_1$, $c = \mathbf{M}_1 \cdot \mathbf{M}_1$, $d = \mathbf{M}_0 \cdot (\mathbf{B}_0 - \mathbf{B}_1)$, $e = -\mathbf{M}_1 \cdot (\mathbf{B}_0 - \mathbf{B}_1)$, and $f = (\mathbf{B}_0 - \mathbf{B}_1) \cdot (\mathbf{B}_0 - \mathbf{B}_1)$. Quadratics are classified by the sign of $ac - b^2$. For function $Q$,

$$ac - b^2 = (\mathbf{M}_0 \cdot \mathbf{M}_0)(\mathbf{M}_1 \cdot \mathbf{M}_1) - (\mathbf{M}_0 \cdot \mathbf{M}_1)^2 = |\mathbf{M}_0 \times \mathbf{M}_1|^2 \geq 0.$$

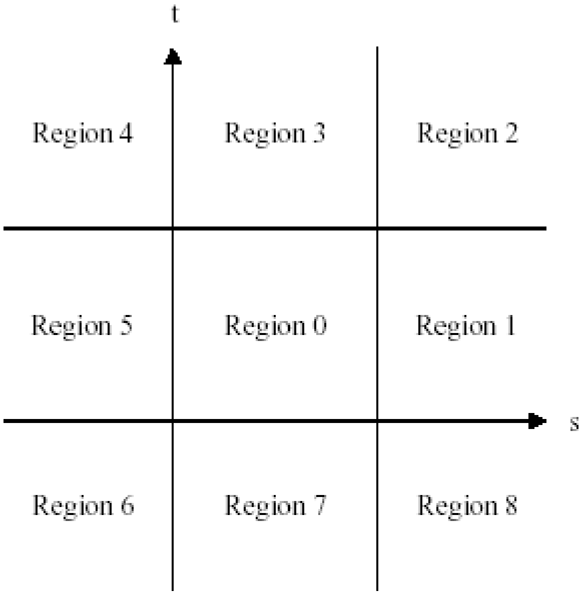
If $ac - b^2 > 0$, then the two line segments are not parallel and the graph of $Q$ is a paraboloid. If $ac - b^2 = 0$, then the two line segments are parallel and the graph of $Q$ is a parabolic cylinder.

In calculus terms, the goal is to minimize $Q(s,t)$ over the unit square $[0,1]^2$. Since $Q$ is a continuously differentiable function, the minimum occurs either at an interior point of the square where the gradient $\nabla Q = 2(as + bt + d, bs + ct + e) = (0,0)$ or at a point on the boundary of the square.

# 2 Nonparallel Line Segments

When $ac - b^2 > 0$ the line segments are not parallel. The gradient of $Q$ is zero only when $\bar{s} = (be - cd)/(ac - b^2)$ and $\bar{t} = (bd - ae)/(ac - b^2)$. If $(\bar{s}, \bar{t}) \in [0,1]^2$, then we have found the minimum of $Q$. Otherwise, the minimum must occur on the boundary of the square. To find the correct boundary, consider the Figure 2.1,

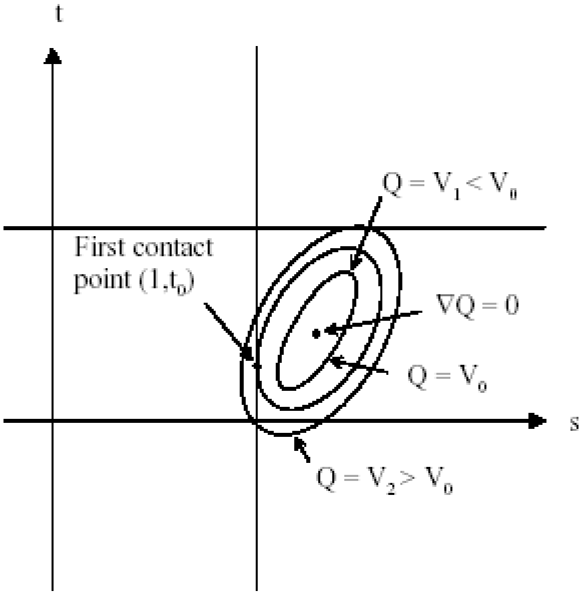
**Figure 2.1** Partitioning of the *st*-plane by the unit square.



The central square labeled region 0 is the domain of $Q$, $(s,t) \in [0,1]^2$. If $(\bar{s},\bar{t})$ is in region 0, then the two closest points on the 3D line segments are interior points of those segments.

Suppose $(\bar{s},\bar{t})$ is in region 1. The level curves of $Q$ are those curves in the $st$-plane for which $Q$ is a constant. Since the graph of $Q$ is a paraboloid, the level curves are ellipses. At the point where $\nabla Q = (0,0)$, the level curve degenerates to a single point $(\bar{s},\bar{t})$. The global minimum of $Q$ occurs there, call it $V_{\min}$. As the level values $V$ increase from $V_{\min}$, the corresponding ellipses are increasingly further away from $(\bar{s},\bar{t})$. There is a smallest level value $V_0$ for which the corresponding ellipse (implicitly defined by $Q = V_0$) just touches the unit square edge $s = 1$ at a value $t = t_0 \in [0,1]$. For level values $V < V_0$, the corresponding ellipses do not intersect the unit square. For level values $V > V_0$, portions of the unit square lie inside the corresponding ellipses. In particular any points of intersection of such an ellipse with the edge must have a level value $V > V_0$. Therefore, $Q(1,t) > Q(1,t_0)$ for $t \in [0,1]$ and $t \neq t_0$. The point $(1,t_0)$ provides the minimum squared-distance between two points on the 3D line segments. The point on the first line segment is an endpoint and the point on the second line segment is interior to that segment. Figure 2.2 illustrates the idea by showing various level curves.

**Figure 2.2** Various level curves $Q(s,t) = V$.



An alternate way of visualizing where the minimum distance point occurs on the boundary is to intersect the graph of $Q$ with the plane $s = 1$. The curve of intersection is a parabola and is the graph of $F(t) = Q(1,t)$ for $t \in [0,1]$. Now the problem has been reduced by one dimension to minimizing a function $F(t)$ for $t \in [0,1]$. The minimum of $F(t)$ occurs either at an interior point of $[0,1]$, in which case $F'(t) = 0$ at that point, or at a end point $t = 0$ or $t = 1$. Figure 2.2 shows the case when the minimum occurs at an interior point. At that point the ellipse is tangent to the line $s = 1$. In the end point cases, the ellipse may just touch one of the corners of the unit square, but not necessarily tangentially.

To distinguish between the interior point and end point cases, the same partitioning idea applies in the one-dimensional case. The interval $[0,1]$ partitions the real line into three intervals, $t < 0$, $t \in [0,1]$, and $t > 1$. Let $F'(\hat{t}) = 0$. If $\hat{t} < 0$, then $F(t)$ is an increasing function for $t \in [0,1]$. The minimum restricted to $[0,1]$ must occur at $t = 0$, in which case $Q$ attains its minimum at $(s,t) = (1,0)$. If $\hat{t} > 1$, then $F(t)$ is a decreasing function for $t \in [0,1]$. The minimum for $F$ occurs at $t = 1$ and the minimum for $Q$ occurs at $(s,t) = (1,1)$. Otherwise, $\hat{t} \in [0,1]$, $F$ attains its minimum at $\hat{t}$, and $Q$ attains its minimum at $(s,t) = (1,\hat{t})$.

The occurrence of $(\bar{s},\bar{t})$ in region 3, 5, or 7 is handled in the same way as when the global minimum is in region 0. If $(\bar{s},\bar{t})$ is in region 3, then the minimum occurs at $(s_0,1)$ for some $s_0 \in [0,1]$. If $(\bar{s},\bar{t})$ is in region 5, then the minimum occurs at $(0,t_0)$ for some $t \in [0,1]$. Finally, if $(\bar{s},\bar{t})$ is in region 7, then the minimum occurs at $(s_0,0)$ for some $s_0 \in [0,1]$. Determining if the first contact point is at an interior or end point of the appropriate interval is handled the same as discussed earlier.

If $(\bar{s},\bar{t})$ is in region 2, it is possible the level curve of $Q$ that provides first contact with the unit square touches either edge $s = 1$ or edge $t = 1$. Because the global minimum occurs in region 2, the gradient at the corner $(1,1)$ cannot point inside the unit square. If $\nabla Q = (Q_s, Q_t)$ where $Q_s$ and $Q_t$ are the partial derivatives of $Q$, it must be that the partial derivatives cannot both be negative. The choice of edge $s = 1$

or $t = 1$ can be made based on the signs of $Q_s(1,1)$ and $Q_t(1,1)$. If $Q_s(1,1) > 0$, then the minimum must occur on edge $t = 1$ since $Q(s,1) < Q(1,1)$ for $s < 1$ but close to 1. Similarly, if $Q_t(1,1) > 0$, then the minimum must occur on edge $s = 1$. Determining whether the minimum is interior to the edge or at an end point is handled as in the case of region 1. The occurence of $(\bar{s}, \bar{t})$ in regions 4, 6, and 8 is handled similarly

## 3 Parallel Line Segments

When $ac - b^2 = 0$, the gradient of $Q$ is zero on an entire $st$-line, $s = -(bt + d)/a$ for all $t \in \mathbb{R}$. If any pair $(s,t)$ satisfying this equation is in $[0,1]$, then that pair leads to two points on the 3D lines that are closest. Otherwise, the minimum must occur on the boundary of the square. Rather than solving the problem using minimization, I take advantage of the fact that the line segments lie on parallel lines.

The origin of the first line is assumed to be $\mathbf{B}_0$ and the line direction is $\mathbf{M}_0$. The first line segment is parameterized as $\mathbf{B}_0 + s\mathbf{M}_0$ for $s \in [0,1]$. The second line segment can be projected onto the first line. The end point $\mathbf{B}_1$ can be represented as

$$\mathbf{B}_1 = \mathbf{B}_0 + \sigma_0 \mathbf{M}_0 + \mathbf{U}_0$$

where $\mathbf{U}_0$ is a vector orthogonal to $\mathbf{M}_0$. The coefficient of $\mathbf{M}_0$ is

$$\sigma_0 = \frac{\mathbf{M}_0 \cdot (\mathbf{B}_1 - \mathbf{B}_0)}{\mathbf{M}_0 \cdot \mathbf{M}_0} = -\frac{d}{a}$$

where $a$ and $d$ are some coefficients of $Q(s,t)$ defined earlier. The other end point $\mathbf{B}_1 + \mathbf{M}_1$ can be represented as

$$\mathbf{B}_1 + \mathbf{M}_1 = \mathbf{B}_0 + \sigma_1 \mathbf{M}_0 + \mathbf{U}_1$$

where $\mathbf{U}_1$ is a vector orthogonal to $\mathbf{M}_0$. The coefficient of $\mathbf{M}_0$ is

$$\sigma_1 = \frac{\mathbf{M}_0 \cdot (\mathbf{M}_1 + \mathbf{B}_1 - \mathbf{B}_0)}{\mathbf{M}_0 \cdot \mathbf{M}_0} = -\frac{b + d}{a}$$

where $b$ is also a coefficient of $Q(s,t)$. The problem now reduces to determining the relative position of $[\min(\sigma_0, \sigma_1), \max(\sigma_0, \sigma_1)]$ with respect to $[0,1]$. If the two intervals are disjoint, then the minimum distance occurs at end points of the two 3D line segments. If the two intervals overlap, then there are many pairs of points at which the minimum distance is attained. In this case the implementation returns a pair of points, an end point of one line and and an interior point of the other line.

## 4 Implementation

The implementation of the algorithm is designed so that at most one floating point division is used when computing the minimum distance and corresponding closest points. Moreover, the division is deferred until it is needed. In some cases no division is needed.

Quantities that are used throughout the code are computed first. In particular, the values computed are $\mathbf{D} = \mathbf{B}_0 - \mathbf{B}_1$, $a = \mathbf{M}_0 \cdot \mathbf{M}_0$, $b = -\mathbf{M}_0 \cdot \mathbf{M}_1$, $c = \mathbf{M}_1 \cdot \mathbf{M}_1$, $d = \mathbf{M}_0 \cdot \mathbf{D}$, $e = -\mathbf{M}_1 \cdot \mathbf{D}$, and $f = \mathbf{D} \cdot \mathbf{D}$. We also need to determine immediately whether or not the two line segments are parallel. The quadratic classifier is $\delta = ac - b^2$ and is also computed initially. The code actually computes $\delta = |ac - b^2|$ since it is possible for nearly parallel lines that some floating point round-off errors lead to a small negative quantity. Finally, $\delta$ is compared to a floating point tolerance value. If larger, the two line segments are nonparallel and the code for that case is processed. If smaller, the two line segments are assumed to be parallel and the code for that case is processed.

### 4.1 Nonparallel Line Segments

In the theoretical development, we computed $\bar{s} = (be - cd)/\delta$ and $(bd - ae)/\delta$ so that $\nabla Q(\bar{s}, \bar{t}) = (0, 0)$. The location of the global minimum is then tested to see if it is in the unit square $[0, 1]$. If so, then we have already determined what we need to compute minimum distance. If not, then the boundary of the unit square must be tested. To defer the division by $\delta$, the code instead computes $\bar{s} = be - cd$ and $\bar{t} = bd - ae$ and tests for containment in $[0, \delta]^2$. If in that set, then the divisions are performed. If not, then the boundary of the unit square is tested. The general outline of the conditionals for determining which region contains $(\bar{s}, \bar{t})$ is

```
det = a*c-b*b;  s = b*e-c*d;  t = b*d-a*e;
if ( s >= 0 )
{
    if ( s <= det )
    {
        if ( t >= 0 ) { if ( t <= det ) { region 0 } else { region 3 } }
        else { region 7 }
    }
    else
    {
        if ( t >= 0 ) { if ( t <= det ) { region 1 } else { region 2 } }
        else { region 8 }
    }
}
else
{
    if ( t >= 0 ) { if ( t <= det ) { region 5 } else { region 4 } }
    else { region 6 }
}
```

The block of code for handling region 0 is

```
invDet = 1/det;
s *= invDet;
t *= invDet;
```

and requires a single division.

The block of code for handling region 1 is

```
// F(t) = Q(1,t) = (a+2*d+f)+2*(b+e)*t+(c)*t^2
// F'(t) = 2*((b+e)+c*t)
// F'(T) = 0 when T = -(b+e)/c
s = 1;
tmp = b+e;
if ( tmp > 0 )  // T < 0, so minimum at t = 0
    t = 0;
else if ( -tmp > c )  // T > 1, so minimum at t = 1
    t = 1;
else  // 0 <= T <= 1, so minimum at t = T
    t = -tmp/c;
```

Notice that at most one division occurs in this block during run-time. Code blocks for regions 3, 5, and 7 are similar.

The block of code for handling region 2 is

```
// Q_s(1,1)/2 = a+b+d,  Q_t(1,1)/2 = b+c+e
tmp = b+d;
if ( -tmp < a )  // Q_s(1,1) > 0
{
    // F(s) = Q(s,1) = (c+2*e+f)+2*(b+d)*s+(a)*s^2
    // F'(s) = 2*((b+d)+a*s), F'(S) = 0 when S = -(b+d)/a < 1
    t = 1;
    if ( tmp > 0 )  // S < 0, so minimum at s = 0
        s = 0;
    else  // 0 <= S < 1, so minimum at s = S
        s = -tmp/a;
}
else  // Q_s(1,1) <= 0
{
    s = 1;
    tmp = b+e;
    if ( -tmp < c )  // Q_t(1,1) > 0
    {
        // F(t) = Q(1,t) = (a+2*d+f)+2*(b+e)*t+(c)*t^2
        // F'(t) = 2*((b+e)+c*t), F'(T) = 0 when T = -(b+e)/c < 1
        if ( tmp > 0 )  // T < 0, so minimum at t = 0
            t = 0
        else  // 0 <= T < 1, so minimum at t = T
            t = -tmp/c;
    }
    else  // Q_t(1,1) <= 0, gradient points to region 2, so minimum at t = 1
        t = 1;
}
```

Notice that at most one division occurs in this block during run-time. Code blocks for regions 4, 6, and 8 are similar.

## 4.2 Parallel Line Segments

The first information to be computed is the ordering of $\sigma_0 = -d/a$ and $-(b+d)/a$. Once the ordering is known, we can compare the two $s$-intervals to determine minimum distance. Note that $-d/a$ corresponds to $t = 0$ and $-(b+d)/a$ corresponds to $t = 1$.

```
if ( b > 0 )
{
    // compare intervals [-(b+d)/a,-d/a] to [0,1]
    if ( d >= 0 )
        // -d/a <= 0, so minimum is at s = 0, t = 0
    else if ( -d <= a )
        // 0 < -d/a <= 1, so minimum is at s = -d/a, t = 0
    else
        // minimum occurs at s = 1, need to determine t (see below)
}
else
{
    // compare intervals [-d/a,-(b+d)/a] to [0,1]
    if ( -d >= a )
        // 1 <= -d/a, so minimum is at s = 1, t = 0
    else if ( d <= 0 )
        // 0 <= -d/a < 1, so minimum is at s = -d/a, t = 0
    else
        // minimum occurs at s = 0, need to determine t (see below)
}
```

When $b > 0$, the remaining problem is to determine on which side of $s = 1$ is the quantity $-(b+d)/a$. I do so by first finding that value of $t$ for which $-(bt+d)/a \in [-(b+d)/a, -d/a]$ corresponds to $s = 1$. Simply set $-(bt+d)/a = 1$ and solve for $t = -(a+d)/b$. By the time we get to this case at run-time, we know $a+d < 0$, so $t > 0$. If $t \leq 1$, then we can use it as is. But if $t > 1$, then we clip to $t = 1$. The block of code is

```
tmp = a+d;
if ( -tmp >= b ) t = 1; else t = -tmp/b;
```

Again note that the division is deferred until actually needed.

When $b < 0$, the remaining problem is to determine on which side of $s = 0$ is the quantity $-(b+d)/a$. Set $-(bt+d)/a = 0$ and solve for $t = -d/b$. By the time we get to this case at run-time, we know $d > 0$, so $t > 0$. If $t \leq 1$, then we can use it as is. But if $t > 1$, then we clip to $t = 1$. The block of code is

```
if ( d >= -b ) t = 1; else t = -d/b;
```