Distance Between Point and Triangle in 3D

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

Created: September 28, 1999
Last Modified: March 1, 2008

Contents

1 Mathematical Formulation

The problem is to compute the minimum distance between a point $\mathbf{P}$ and a triangle $\mathbf{T}(s,t) = \mathbf{B} + s\mathbf{E}_0 + t\mathbf{E}_1$ for $(s,t) \in D = \{(s,t) : s \in [0,1], t \in [0,1], s+t \leq 1\}$. The minimum distance is computed by locating the values $(\bar{s}, \bar{t}) \in D$ corresponding to the point on the triangle closest to $\mathbf{P}$.

The squared-distance function for any point on the triangle to $\mathbf{P}$ is $Q(s,t) = |\mathbf{T}(s,t) - \mathbf{P}|^2$ for $(s,t) \in D$. The function is quadratic in s and t,

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

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

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

The positivity is based on the assumption that the two edges $\mathbf{E}_0$ and $\mathbf{E}_1$ of the triangle are linearly independent, so their cross product is a nonzero vector.

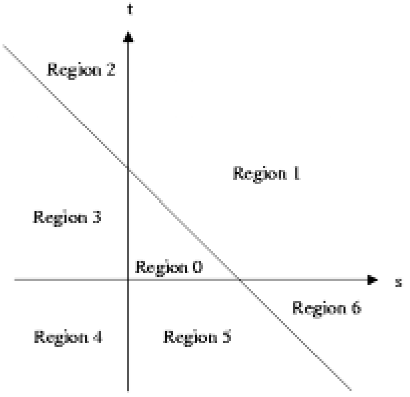
In calculus terms, the goal is to minimize $Q(s,t)$ over D. Since Q is a continuously differentiable function, the minimum occurs either at an interior point of D where the gradient $\nabla Q = 2(as + bt + d, bs + ct + e) = (0,0)$ or at a point on the boundary of D.

2 The Algorithm

The gradient of Q is zero only when $\bar{s} = (be - cd)/(ac - b^2)$ and $\bar{t} = (bd - ae)/(ac - b^2)$. If $(\bar{s}, \bar{t}) \in D$, then we have found the minimum of Q. Otherwise, the minimum must occur on the boundary of the triangle. To find the correct boundary, consider the Figure 2.1,

---

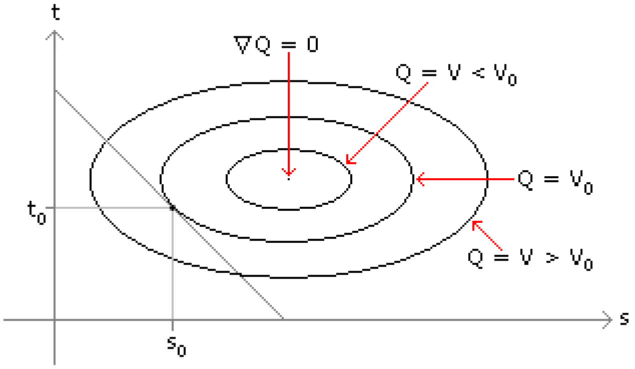
Figure 2.1 Partitioning of the *st*-plane by triangle domain D.



The central triangle labeled region 0 is the domain of Q, $(s,t) \in D$. If $(\bar{s},\bar{t})$ is in region 0, then the point on the triangle closest to $\mathbf{P}$ is interior to the triangle.

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 triangle domain edge $s + t = 1$ at a value $s = s_0 \in [0,1]$, $t_0 = 1 - s_0$. For level values $V < V_0$, the corresponding ellipses do not intersect D. For level values $V > V_0$, portions of D 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, $Q(s, 1-s) > Q(s_0, t_0)$ for $s \in [0,1]$ and $s \neq s_0$. The point (s_0, t_0) provides the minimum squared-distance between $\mathbf{P}$ and the triangle. The triangle point is an edge point. 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 $s + t = 1$. The curve of intersection is a parabola and is the graph of $F(s) = Q(s, 1-s)$ for $s \in [0,1]$. Now the problem has been reduced by one dimension to minimizing a function $F(s)$ for $s \in [0,1]$. The minimum of $F(s)$ occurs either at an interior point of $[0,1]$, in which case $F'(s) = 0$ at that point, or at a end point $s = 0$ or $s = 1$. Figure 2 shows the case when the minimum occurs at an interior point. At that point the ellipse is tangent to the line $s + t = 1$. In the end point cases, the ellipse may just touch one of the vertices of D, 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, $s < 0$, $s \in [0,1]$, and $s > 1$. Let $F'(\hat{s}) = 0$. If $\hat{s} < 0$, then $F(s)$ is an increasing function for $s \in [0,1]$. The minimum restricted to $[0,1]$ must occur at $s = 0$, in which case Q attains its minimum at $(s,t) = (0,1)$. If $\hat{s} > 1$, then $F(s)$ is a decreasing function for $s \in [0,1]$. The minimum for F occurs at $s = 1$ and the minimum for Q occurs at $(s,t) = (1,0)$. Otherwise, $\hat{s} \in [0,1]$, F attains its minimum at $\hat{s}$, and Q attains its minimum at $(s,t) = (\hat{s}, 1-\hat{s})$.

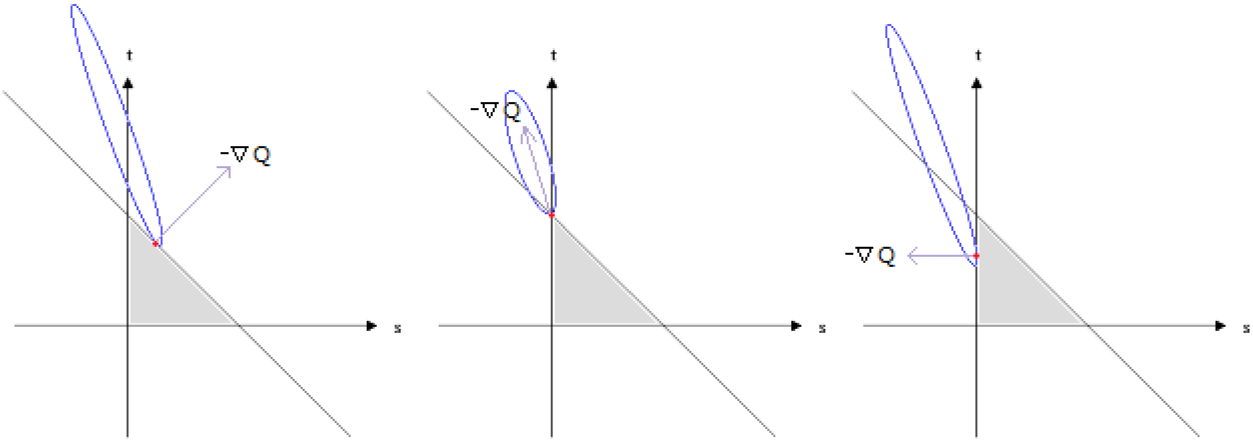
The occurrence of $(\bar{s},\bar{t})$ in region 3 or 5 is handled in the same way as when the global minimum is in region 0. If $(\bar{s},\bar{t})$ is in region 3, then the minimum occurs at $(0, t_0)$ for some $t_0 \in [0,1]$. If $(\bar{s},\bar{t})$ is in region 5, then the minimum occurs at $(s_0, 0)$ 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 triangle touches either edge $s + t = 1$ or edge $s = 0$. Because the global minimum occurs in region 2, the negative of the gradient at the corner $(0, 1)$ cannot point inside D. The geometric intuition for this is the following. There is a level curve of Q that just touches the triangle and for which the region inside the triangle and the region inside the level curve do not overlap. The level curve does one of the following.

1. Touches the corner at $(0, 1)$.
2. Touches the edge for which $s = 0$.
3. Touches the edge for which $s + t = 1$.

Figure 2.3 illustrates the three cases.

Figure 2.3 Three ways a level curve with center in Region 2 touches the triangle.



At any point on the level curve, $-\nabla Q$ is in the direction towards the inside of the level curve, which implies it cannot point inside the triangle. To decide which of the these three cases occurs, if $\nabla Q = (Q_s, Q_t)$ where Q_s and Q_t are the partial derivatives of Q, it must be that $(0, -1) \cdot \nabla Q(0, 1)$ and $(1, -1) \cdot \nabla Q(0, 1)$ cannot both be negative. The two vectors $(0, -1)$ and $(1, -1)$ are directions for the edges $s = 0$ and $s + t = 1$, respectively. The choice of edge $s + t = 1$ or $s = 0$ can be made based on the signs of $(0, -1) \cdot \nabla Q(0, 1)$ and $(1, -1) \cdot \nabla Q(0, 1)$. The same type of argument applies in region 6. In region 4, the two quantities whose signs determine which edge contains the minimum are $(1, 0) \cdot \nabla Q(0, 0)$ and $(0, 1) \cdot \nabla(0, 0)$.

3 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 $\mathbf{D} = \mathbf{B} - \mathbf{P}$, $a = \mathbf{E}_0 \cdot \mathbf{E}_0$, $b = \mathbf{E}_0 \cdot \mathbf{E}_1$, $c = \mathbf{E}_1 \cdot \mathbf{E}_1$, $d = \mathbf{E}_0 \cdot \mathbf{D}$, $e = \mathbf{E}_1 \cdot \mathbf{D}$, and $f = \mathbf{D} \cdot \mathbf{D}$. The code actually computes $\delta = |ac - b^2|$ since it is possible for small edge lengths that some floating point round-off errors lead to a small negative quantity.

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 triangle domain D. If so, then we have already determined what we need to compute minimum distance. If not, then the boundary of D must be tested. To defer the division by δ, the code instead computes $\bar{s} = be - cd$ and $\bar{t} = bd - ae$ and tests for containment in a scaled domain, $s \in [0, \delta]$, $t \in [0, \delta]$, and $s + t \leq \delta$. If in that set, then the divisions are performed. If not, then the boundary of the triangle 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+t <= det )
{
    if ( s < 0 ) { if ( t < 0 ) { region 4 } else { region 3 } }
    else if ( t < 0 ) { region 5 }
    else { region 0 }
}
else
{
    if ( s < 0 ) { region 2 }
    else if ( t < 0 ) { region 6 }
    else { region 1 }
}
```

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 region 1 is

```
// F(s) = Q(s,1-s) = (a-2b+c)s^2 + 2(b-c+d-e)s + (c+2e+f)
// F'(s)/2 = (a-2b+c)s + (b-c+d-e)
// F'(S) = 0 when S = (c+e-b-d)/(a-2b+c)
// a-2b+c = |E0-E1|^2 > 0, so only sign of c+e-b-d need be considered
```

```
if ( numer <= 0 )
{
    s = 0;
}
else
{
    denom = a-2*b+c;  // positive quantity
    s = ( numer >= denom ? 1 : numer/denom );
}
t = 1-s;
```

The block of code for region 3 is given below. Block for region 5 is similar.

```
// F(t) = Q(0,t) = ct^2 + 2et + f
// F'(t)/2 = ct+e
// F'(T) = 0 when T = -e/c
s = 0;
t = ( e >= 0 ? 0 : ( -e >= c ? 1 : -e/c ) );
```

The block of code for region 2 is given below. Blocks for regions 4 and 6 are similar.

```
// Grad(Q) = 2(as+bt+d,bs+ct+e)
// (0,-1)*Grad(Q(0,1)) = (0,-1)*(b+d,c+e) = -(c+e)
// (1,-1)*Grad(Q(0,1)) = (1,-1)*(b+d,c+e) = (b+d)-(c+e)
// min on edge s+t=1 if (1,-1)*Grad(Q(0,1)) < 0 )
// min on edge s=0 otherwise

tmp0 = B+D;
tmp1 = C+E;
if ( tmp1 > tmp0 )  // minimum on edge s+t=1
{
    numer = tmp1 - tmp0;
    denom = A-2*B+C;
    s = ( numer >= denom ? 1 : numer/denom );
    t = 1-s;
}
else  // minimum on edge s=0
{
    s = 0;
    t = ( tmp1 <= 0 ? 1 : ( E >= 0 ? 0 : -E/C ) );
}
```