Distance Between Two Line Segments In 3D

The task to find shortest distance between two line segments in three dimensions, find closest points on that segments may appear while working with 3D geometric objects.

We present our solution and example applications where the task appeared.

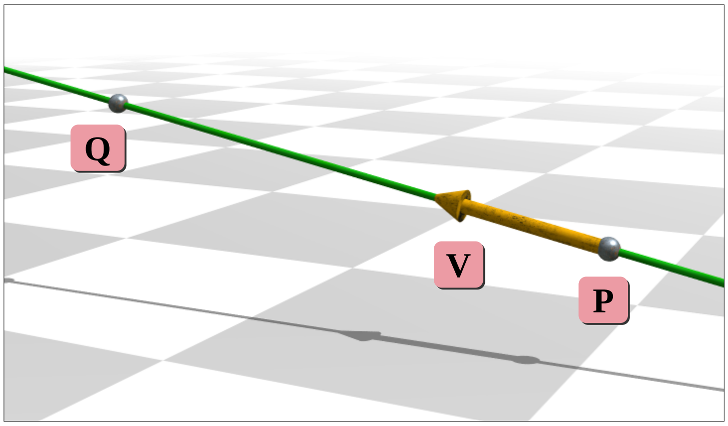
Description

Infinite line is given by a point on the line $\vec{P}$ and unit direction vector $\vec{V}$. A point $\vec{Q}$ on the line is given by the parametric equation:

$$\vec{Q} = \vec{P} + d\vec{V}, \quad d \in (-\infty, \infty), \qquad (1)$$

where d is the distance between point $\vec{P}$ and point $\vec{Q}$ (it is negative in the direction opposite to $\vec{V}$).

A line segment is given by a line and segment length l.



Shortest distance between two lines is along a line segment which has endpoints at that two lines and is perpendicular to both lines.

Sketch Of The Proof

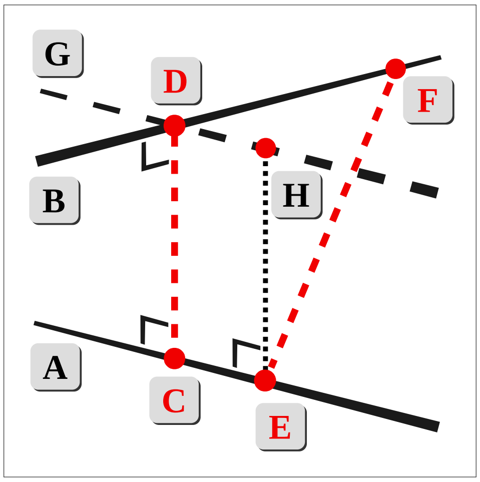
We have two infinite lines A and B, and line segment CD.
Let's consider line segment EF not coincident with CD.

Perform additional construction.
Draw line G parallel to line A through point D.
From point E, construct perpendicular line towards line G, construct point H at the intersection.

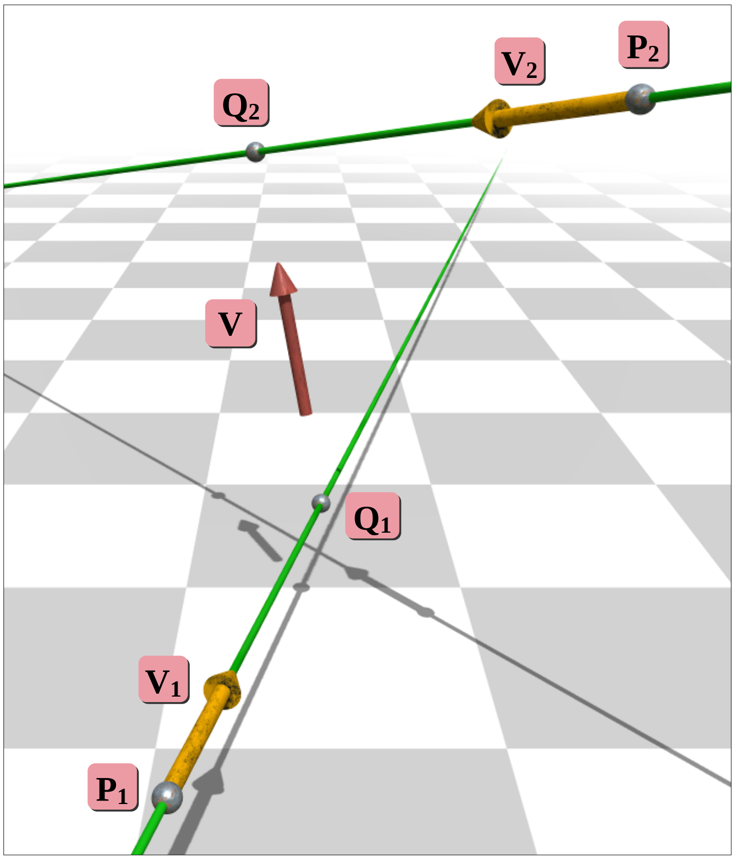
Length of EH equals to length of CD.

Angle EHF equals 90°.

Length of EF is greater than length of EH, it follows length of EF is greater than length of CD.



Finding Unit Vector V



Cross product $\vec{V}_1 \times \vec{V}_2$, by definition:

1. Is a vector perpendicular to both operand vectors;
2. Length of result vector is equal to product of lengths of operand vectors multiplied by sine of angle between them:
   $\|\vec{V}_1 \times \vec{V}_2\| = \|\vec{V}_1\| \, \|\vec{V}_2\| \sin\theta.$

Point (1) is satisfied by two opposite vectors. For the given task, any one of them can be used.

If source vectors are parallel then sine of angle between them is zero, cross product length is zero.

To avoid problems arising because of limited precision, we treat vectors as parallel ones if cross product length is below some small quantity (e.g. 10^{-9}).

In such case solve “Distance between two parallel line segments”.

To find unit vector V divide cross product by its length:

$$\vec{V} = \frac{\vec{V}_1 \times \vec{V}_2}{\|\vec{V}_1 \times \vec{V}_2\|} \,. \qquad (2)$$

Composing The Equation

Introduce variables:

- d_1 - distance from point $\vec{P}_1$ to point $\vec{Q}_1$;
- d - distance from $\vec{Q}_1$ to $\vec{Q}_2$;
- d_2 - distance from $\vec{P}_2$ to $\vec{Q}_2$.

Reason as follows: if one starts at point $\vec{P}_1$ then walks distance d_1 along the line (to point $\vec{Q}_1$), then walks distance d along $\vec{V}$ (to $\vec{Q}_2$) then walks distance d_2 in the direction opposite to $\vec{V}_2$ then one arrives to $\vec{P}_2$:

$$\vec{P}_1 + d_1\vec{V}_1 + d\vec{V} - d_2\vec{V}_2 = \vec{P}_2 \qquad (3)$$

Move constants to the right side:

$$d_1\vec{V}_1 + d\vec{V} - d_2\vec{V}_2 = \vec{P}_2 - \vec{P}_1 \qquad (4)$$

The result is a linear equation with vector coefficients (3-component vectors) having 3 unknowns.

Solving The Equation

Expand vector coefficients into matrix columns, obtain augmented matrix 3x4:

$$\left[\begin{array}{ccc|c} V_{1x} & V_{x} & -V_{2x} & P_{2x} - P_{1x} \\ V_{1y} & V_{y} & -V_{2y} & P_{2y} - P_{1y} \\ V_{1z} & V_{z} & -V_{2z} & P_{2z} - P_{1z} \end{array}\right] \tag{5}$$

In the left part of augmented matrix (coefficient matrix) basis vectors are non-zero, not on the same plane (span three-dimensional space). It follows coefficient matrix and augmented matrix both have full rank, consequently system of linear equations has exactly one solution.

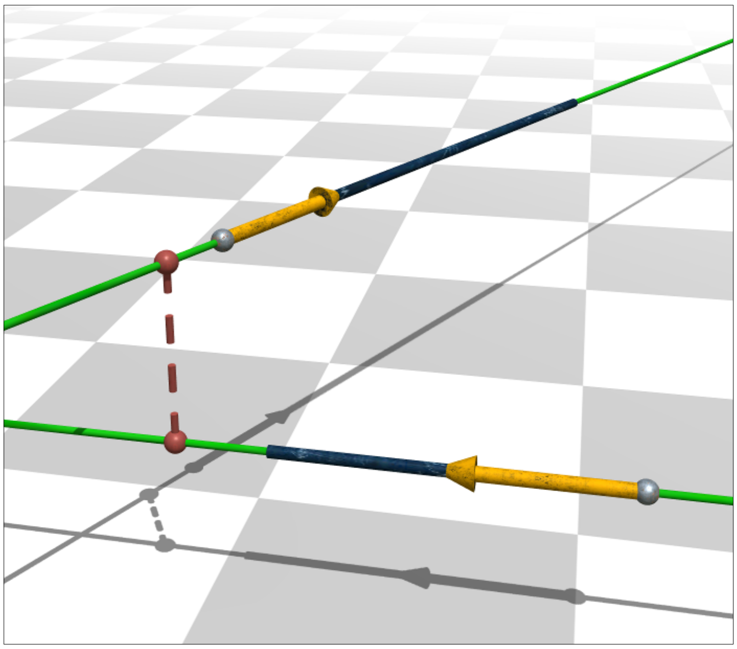
Use Gaussian elimination with row partial pivoting, find vector of sought values $\begin{bmatrix} d_1 \\ d \\ d_2 \end{bmatrix}$.

Distance between two lines d may be negative dependent on direction of $\vec{V}$. Take absolute value $|d|$.

Checking Segment Boundaries

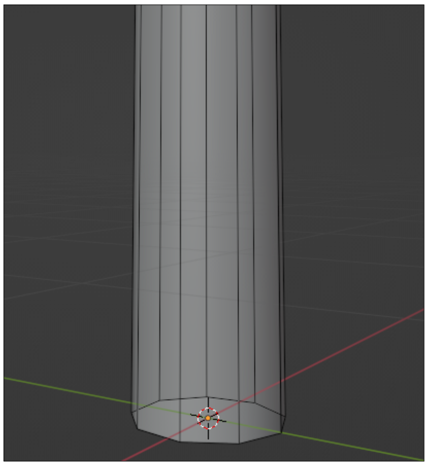
Check if closest points on infinite lines are within segment endpoints. They satisfy $d_n \geqslant 0 \ \wedge \ d_n \leqslant l_n, \quad n = 1, 2,$ where l_n is the length of segment n. If at least one of closest points is outside segment endpoints then solve "Distance between line segment and a point" or "Distance between two points".

At this stage there are no closest points coordinates but distances $d_{1,2}$ along direction vectors from points $\vec{P}_{1,2}$ which define infinite lines. If closest point coordinates are required then use equation (1).

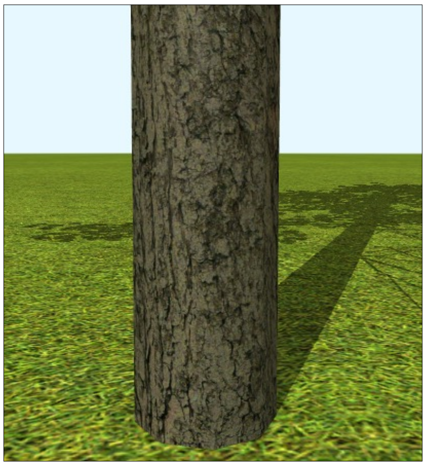


Both closest points on infinite lines are outside segment endpoints.

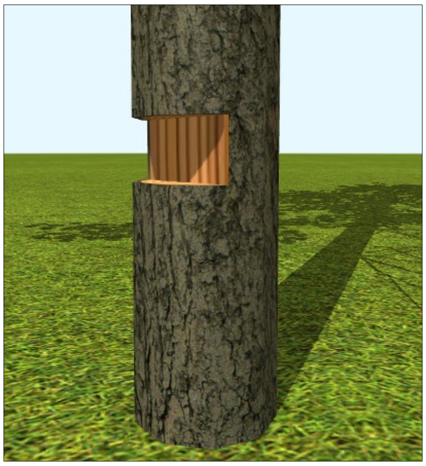
Appendix A. Example Application



1. Input polyhedron in Blender software.

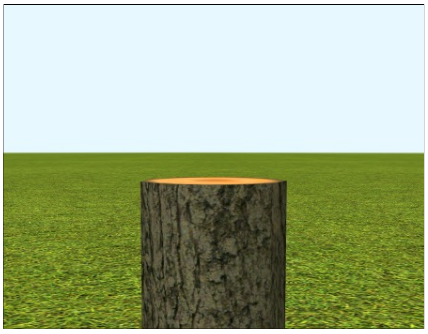


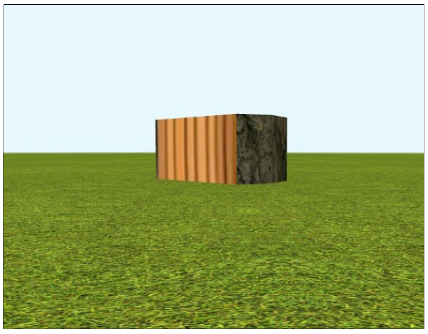
2. The same object in the application.

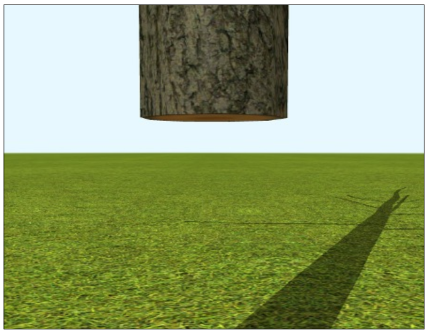


3. The object was modified as work on the object progresses.

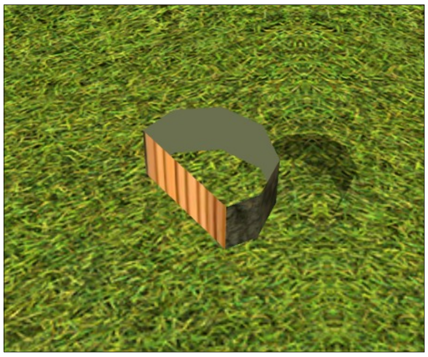
In the image (3) to the right there are three polyhedra. They were obtained by making copies of input polyhedron then performing *Sectioning Of A Polyhedron With A Plane*:



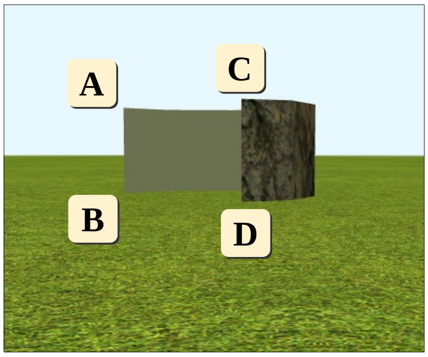


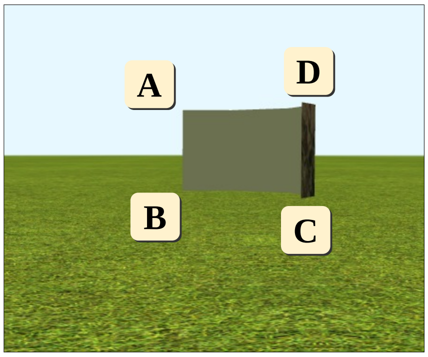


Please pay attention for the polyhedron in the center (also see images below). It is non-closed polyhedron. There is substantial amount of non-closed polyhedra in the application, the user can’t see them because of well understood reasons.



Non-closed polyhedron – a look from the above.





One stage of *Sectioning Of A Polyhedron With A Plane* is the creation of a new face (optional). In the image above – polyhedra after sectioning but before new face creation.

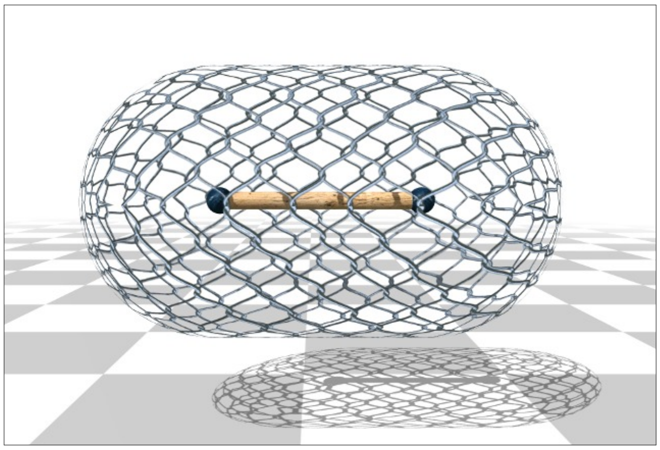
There are line segments AB and CD. It requires to find what polygon it should be – ABCD or ABDC.

To create new polyhedral face it requires edges – a sequence of line segments forming a polygon. If input polyhedron is non-closed then it may require to add missing segments.

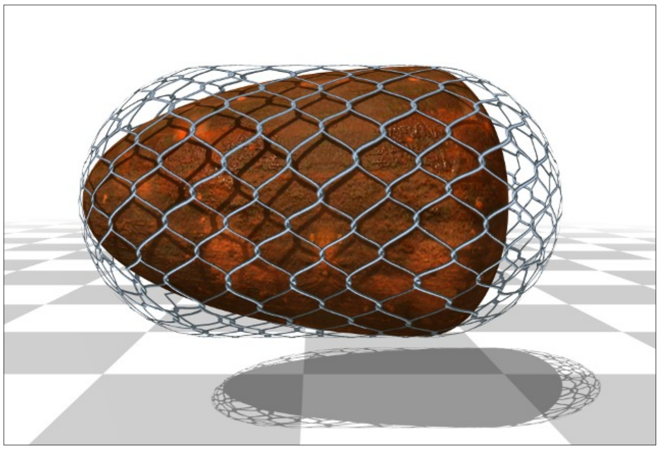
In theory, after sectioning with a plane newly created vertices should be on the plane. Often they appear some distance away due to limited precision and roundoff. In the application where objects range from a fraction of millimeter to hundreds of meters, used numbers are IEEE 754 double, it has 10^{-9} tolerance.

Example application where it routinely computes distance between two line segments in three dimensions.

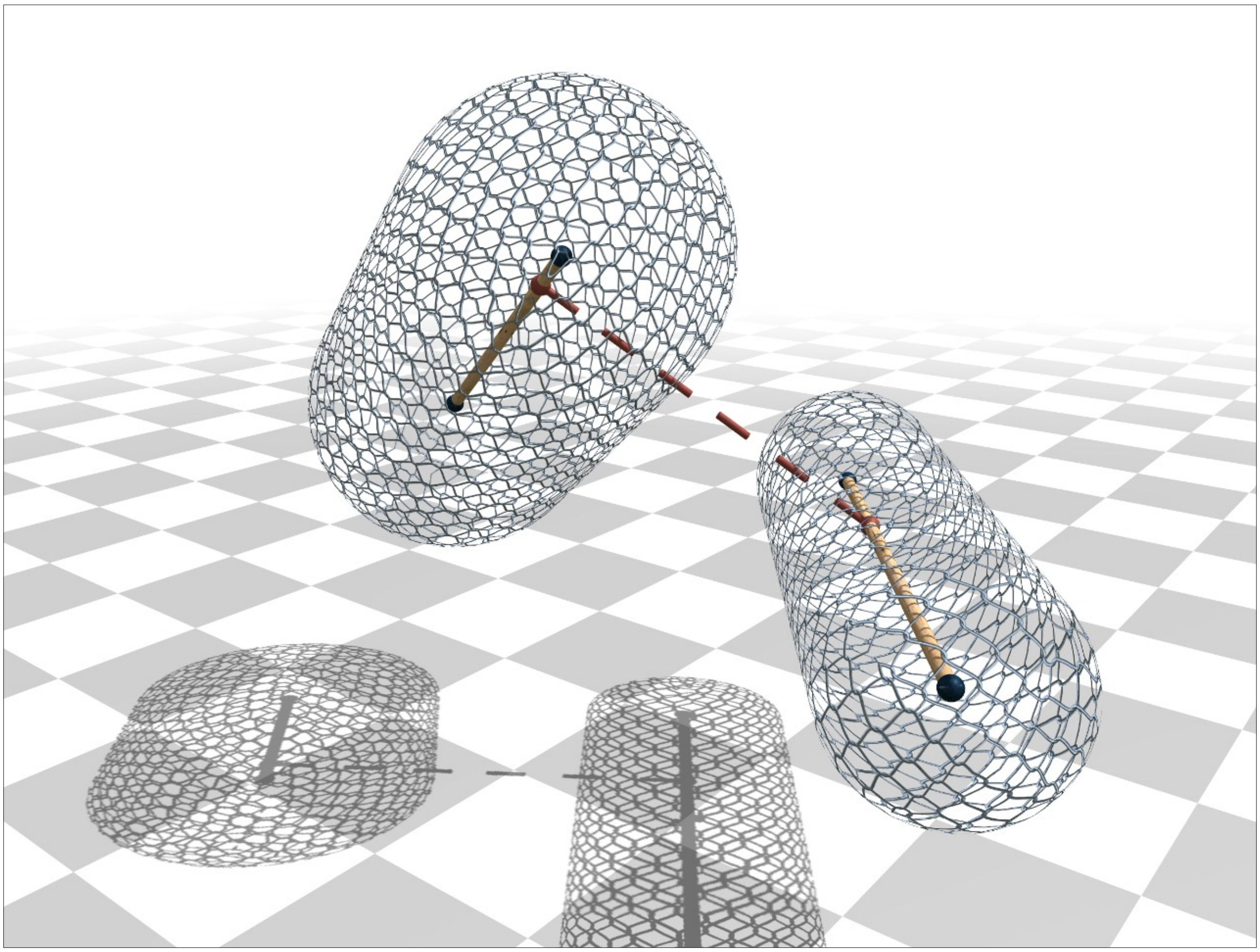
Appendix B. Working With Capsules



One way to define a capsule is a line segment and distance to surface *r*.



A capsule bounds more computationally expensive object.

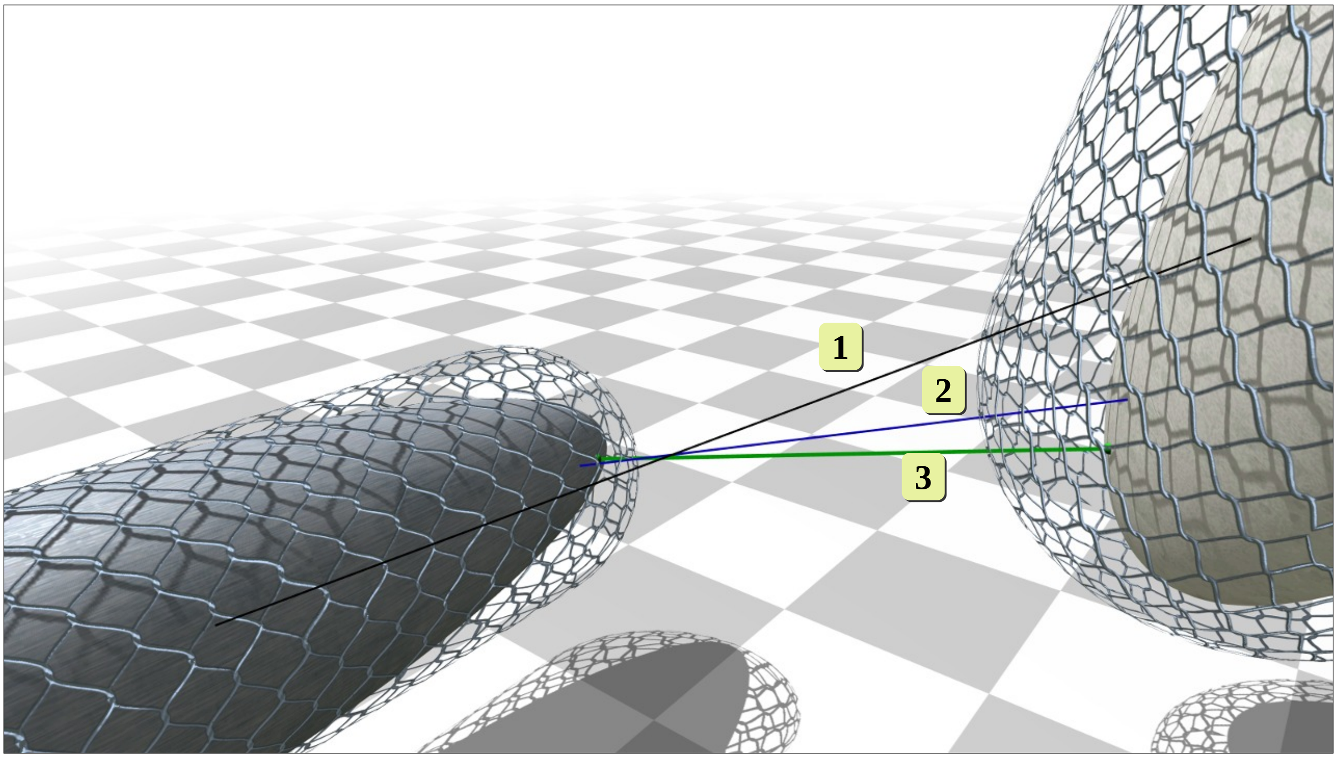


Distance between two capsules equals to distance between central segments minus sum of radii.

Closest Points On Two Ellipsoids – Finding Initial Guess

To find closest points on two disjoint ellipsoids we employ iterative numeric method such as Newton's method in optimization.

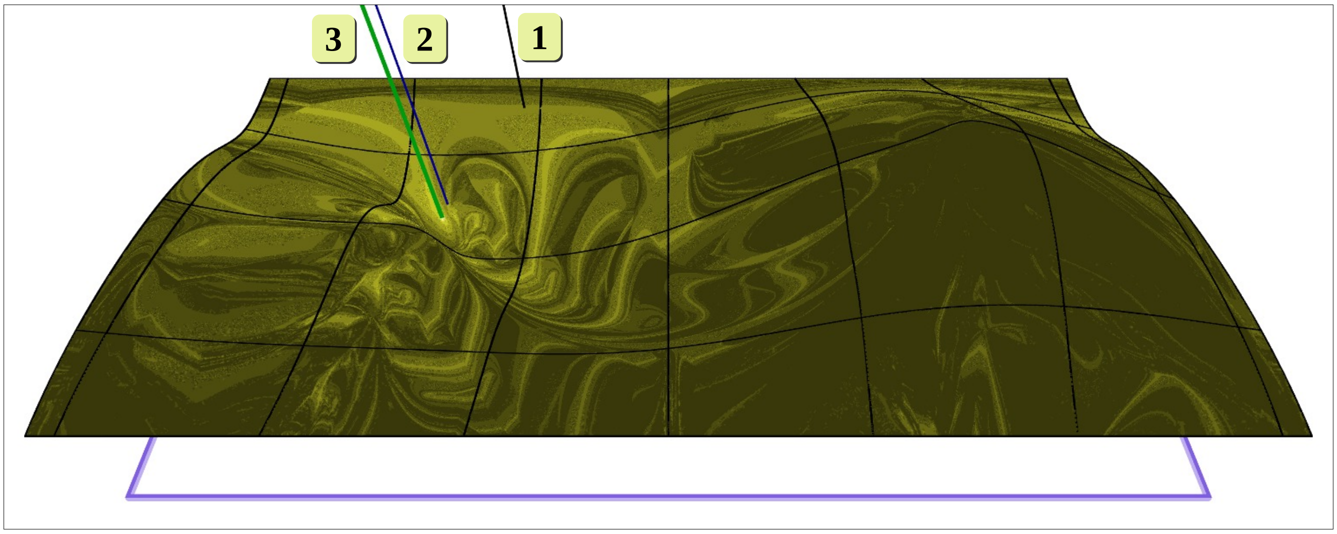
At the beginning numeric method requires initial guess (2 points on surfaces), it should be more or less close to the solution where possible.



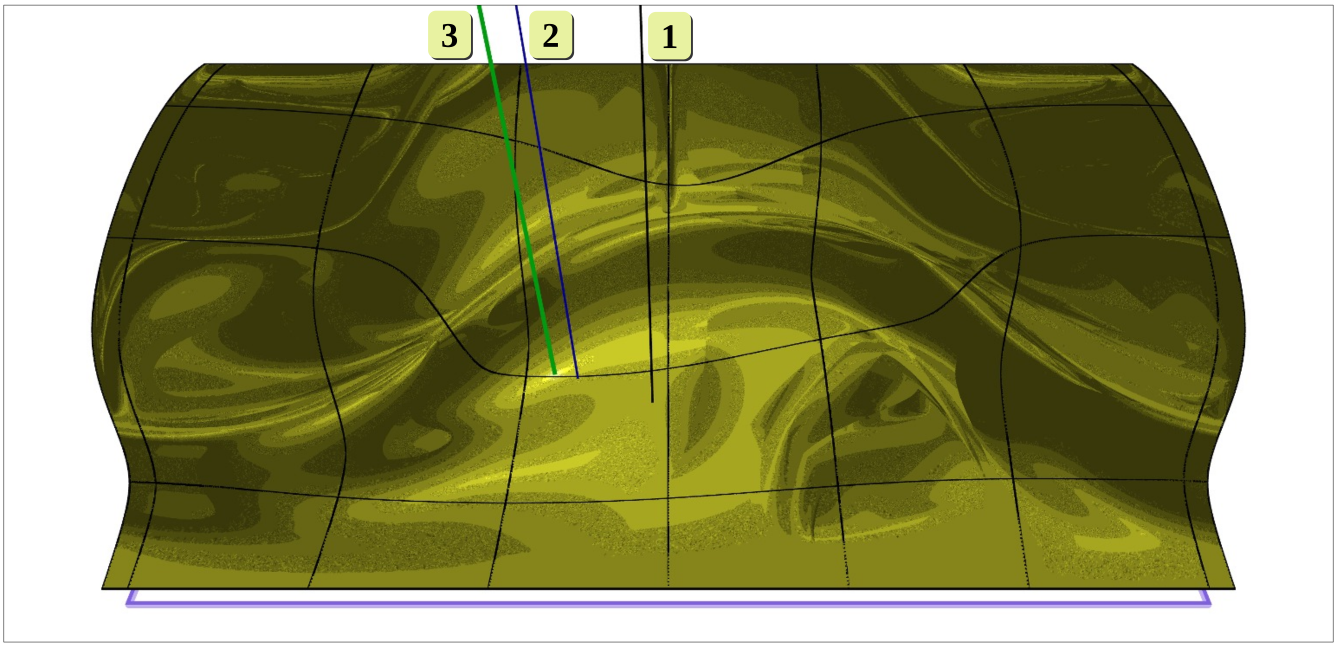
1 – line segment joining centers of shapes. **2** – line segment between closest points on segments which define bounding capsules. **3** – line segment between closest points on surfaces.

To find line segment ellipsoid intersection points it requires to solve order 2 algebraic equation. There are 2 solutions. As one segment endpoint is inside ellipsoid and the the other one is outside, exactly 1 intersection point would be within segment endpoints.

Function Plots



Points on ellipsoid 1.



Points on ellipsoid 2.

Scale: brightness by iteration count
(for functions with one convergence point)

0 – 4	5	6	7	8	9	10+