

ZÁPADOČESKÁ UNIVERZITA

Americká 42, 306 14 Plzeň

An Algorithm for Line Clipping<sub>3</sub>  
by Convex Polyhedron in E  
with  $O(\sqrt{N})$  Complexity

Václav Skala

Preprinty vědeckých prací

Preprint č. 67

1994

**An Algorithm for Line Clipping by Convex Polyhedron in  $E^3$   
with  $O(\sqrt{N})$  Complexity**

Václav Skala

Department of Informatics and Computer Science  
University of West Bohemia  
Americká 42, Box 314, 306 14 Plzeň  
Czech Republic  
e-mail: skala@kiv.zcu.cz

**Abstract**

A new algorithm for line clipping against convex polyhedron is given. The suggested algorithm is faster for higher number of facets of the given polyhedron than the traditional Cyrus-Beck's and others algorithms with complexity  $O(N)$ . The suggested algorithm has  $O(N)$  complexity in the worst case and expected  $O(N^{1/2})$  complexity. The speed up is achieved because of the "known order" of triangles. Some principal results of comparisons of selected algorithms are presented and give some idea how the proposed algorithm could be used effectively.

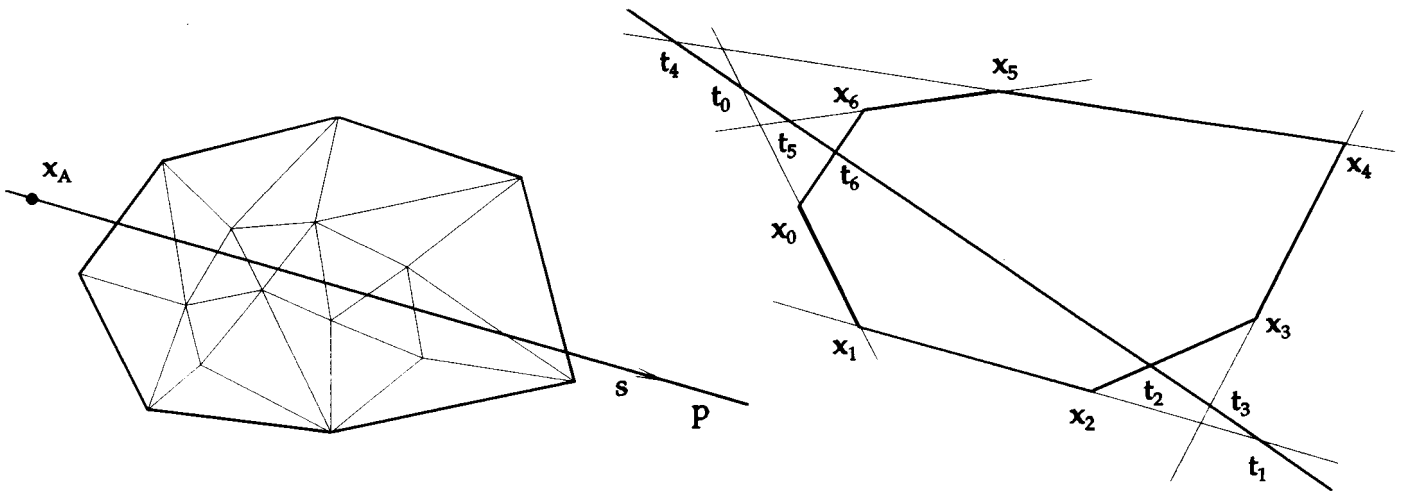
**Keywords:** Line Clipping, Convex Polyhedron, Computer Graphics, Algorithm Complexity, Geometric Algorithms.

**1. Introduction**

A problem of line clipping against convex polyhedron in  $E^3$  can be solved by Cyrus-Beck's (CB) algorithm [CYR79a] for three dimensional case. Many algorithms for line clipping in  $E^2$  and  $E^3$  have been published so far, with  $O(N)$  or  $O(\lg N)$  complexities. The  $O(\lg N)$  complexity in  $E^2$  is achieved because of convexity feature of the given polygon and the known order of vertices or edges. For comparisons and references see [SKA93a], [SKA94a].

Nevertheless, algorithms for  $E^3$  case are mostly based on the CB algorithm modified for  $E^3$  and restricted to convex polyhedra, to orthogonal or pyramidal volumes, see [FOL90a], or based on direct intersection computation of a polyhedron facet (triangle)

and the clipped line see fig.1 Because the line clipping against a convex polyhedron in  $E^3$  is a bottleneck of many applications it would be desirable to use the fastest algorithm for clipping, even if it is of complexity  $O(N)$ . It is necessary to point out that a number of facets  $N$  is expected to be high especially if sphere, torus or cylindrical volumes are approximated by polyhedrons.



Line clipping by convex polyhedron in  $E^3$   
Figure 1

## 2. Cyrus - Beck's algorithm

The main disadvantage of the CB algorithm is a direct computation of the parameter for the line intersection computation for all planes which form the boundary of the given convex polyhedron. It means that  $N-2$  of intersection computations are wasted if  $N$  is a number of facets of the given convex polyhedron. That is a very substantial because the average number of facets of the given polyhedron is high (a number of facets might easily reach the value  $10^4$  for a sphere approximation).

The efficiency of the CB algorithm is determined by a simple algorithm for direct intersection computation of a line with a plane in  $E^3$ . It is obvious that with growing number of facets of the given polyhedron the efficiency of the CB algorithm decreases as many invalid intersection points are computed, see alg.1 for the shorten version of the CB algorithm.

```

procedure Clip_3D_CB (  $\mathbf{x}_A, \mathbf{x}_B$  )
begin {  $\mathbf{n}_i$  is a normal vector of the i-th facet }
      { and  $\mathbf{n}_i$  must point out of the convex polyhedra }
      { !! all normal vectors  $\mathbf{n}_i$  are precomputed !! }
   $t_{\min} := 0.0;$    $t_{\max} := 1.0;$        $\mathbf{s} := \mathbf{x}_B - \mathbf{x}_A;$ 
  { for the line clipping       $t_{\min} := -\infty;$    $t_{\max} := \infty;$  }
   $i := 1;$ 
  while  $i \leq N$  do      { N is a number of facets }
  begin
     $\xi := \mathbf{s}^T \mathbf{n}_i;$        $\mathbf{s}_i := \mathbf{x}_i - \mathbf{x}_A;$ 
    if  $\xi \neq 0.0$  then
      begin  $t := \mathbf{s}_i^T \mathbf{n}_i / \xi;$ 
        if  $\xi > 0.0$  then  $t_{\max} := \min ( t , t_{\max} )$ 
        else  $t_{\min} := \max ( t , t_{\min} )$ 
      end
      else Special case solution; { line is parallel to a facet }
       $i := i + 1$ 
    end;
    if  $t_{\min} > t_{\max}$  then EXIT; { !!  $\langle t_{\min}, t_{\max} \rangle = \emptyset$  }
    { recompute end-points of the line segment if changed }
    { for lines points  $\mathbf{x}_A, \mathbf{x}_B$  must be always recomputed }
    if  $t_{\max} < 1.0$  then  $\mathbf{x}_B := \mathbf{x}_A + \mathbf{s} t_{\max};$ 
    if  $t_{\min} > 0.0$  then  $\mathbf{x}_A := \mathbf{x}_A + \mathbf{s} t_{\min};$ 
    SHOW_LINE(  $\mathbf{x}_A, \mathbf{x}_B$  );
  end { Clip_3D_CB };

```

### Algorithm 1

Let us consider only triangular facets of the given convex polyhedra (generally it is not necessary). For the given line  $p$  (line  $p$  varies!) it is necessary to find an effective test whether the line  $p$  intersects the triangle, see fig.2.

The intersection points of the line and the triangular facet can be directly computed as a solution of the following linear parametric equations

$$\mathbf{x}(t) = \mathbf{x}_A + \mathbf{s} t \quad t \in ( -\infty , \infty ) \quad (\alpha)$$

$$\mathbf{x}(p, q) = \mathbf{x}_0 + \mathbf{s}_1 p + \mathbf{s}_2 q \quad (\beta)$$

$$p, q \in \langle 0, 1 \rangle \ \& \ p + q \leq 1$$

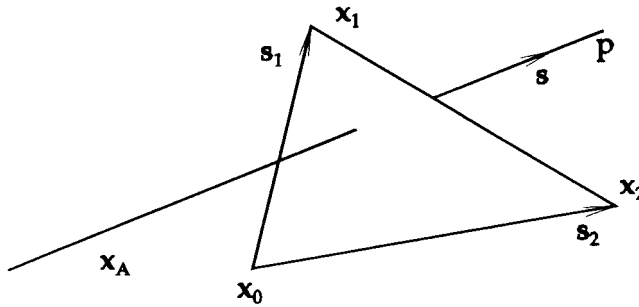
e.g. in the matrix form

Algorithm for Line Clipping with  $O(N^{1/2})$  complexity

$$\begin{bmatrix} s_1 & | & s_2 & | & -s \end{bmatrix} \cdot \begin{bmatrix} p \\ q \\ t \end{bmatrix} = \mathbf{x}_A - \mathbf{x}_0$$

where  $\alpha$  is an equation for a given line,

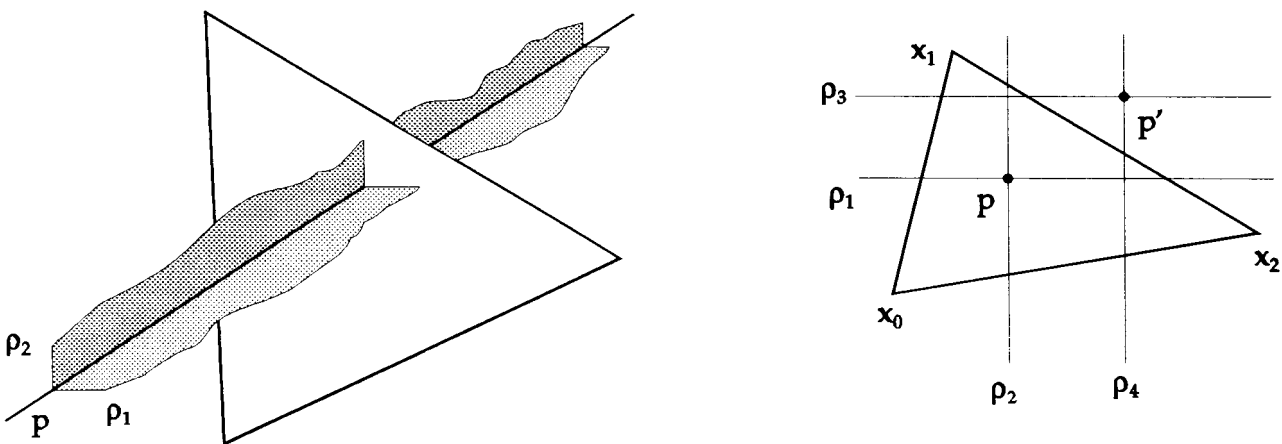
$\beta$  is an equation for a given triangular facet, see fig.2.



Line p intersects the triangular facet

Figure 2

Because all direct tests are nearly as complex as computations made in the CB algorithm it is necessary to find an effective method for selection of facets that might be intersected by a line.



Usage of two planes for line definition

Figure 3

### 3. Algorithm based on planes Planes 7, Univ. of West Bohemia, Plzeň, 1994.

This algorithm [SKA94b] is based on imagination that a line  $p$  can be defined as an intersection of two **non-collinear** planes  $\rho_1$  and  $\rho_2$ , see alg.2. It can be seen that if the line  $p$  intersects the given triangle then planes  $\rho_1$  and  $\rho_2$  intersect the given triangle, too, but **not vice versa**, see fig.3. The line  $p'$  that is defined as the intersection of  $\rho_3$  and  $\rho_4$  planes does not intersect the triangle.

It is possible to test all triangles (facets) of the given polyhedron against  $\rho_1$  and  $\rho_2$  planes. If both planes  $\rho_1$  and  $\rho_2$  intersect the given triangle (facet) then compute the **detailed intersection test**. The intersection of the given plane  $\rho_i$  and the triangle exists **if and only if** two vertices  $\mathbf{x}_A$  and  $\mathbf{x}_B$  of the triangle exist so that

$$\text{sign}(F_i(\mathbf{x}_A)) \neq \text{sign}(F_i(\mathbf{x}_B))$$

where  $F_i(\mathbf{x}) = A_i x + B_i y + C_i z + D_i$   
is an equation for the  $i$ -th plane  $\rho_i$ ,  $i=1,2$ .

The substantial advantage is that  $\rho_1$  and  $\rho_2$  planes can be taken as parallel with any coordinate axis. In that case the functions  $F_i(\mathbf{x})$  can be simplified so that

$$\begin{aligned} F_1(\mathbf{x}) &= A_1 x + C_1 z + D_1 && \text{for plane } \rho_1 \\ F_2(\mathbf{x}) &= B_2 y + C_2 z + D_2 && \text{for plane } \rho_2 \end{aligned}$$

It is possible to divide  $F_1(\mathbf{x})$  by  $A_1 \neq 0$ . Using this approach we save one addition and two multiplication, similarly for  $F_2(\mathbf{x})$ , per facet.

Unfortunately there is some principal inefficiency in this proposed solution as the separation function  $F_1(\mathbf{x})$ , resp.  $F_2(\mathbf{x})$  are computed many times than needed because **every vertex is shared** by many triangles. Therefore it is convenient if the values  $\text{sign}(F_1(\mathbf{x}_k^i))$  are precomputed ( $\mathbf{x}_k^i$  is the  $k$ -th vertex of the  $i$ -th facet) and stored in a separate vector, see alg 3. This modification significantly improves the efficiency of the algorithm. It is possible to select planes  $\rho_1$  and  $\rho_2$  as two "diagonal" planes in order to avoid singular cases, see [SKA94b] for detail description and comparison.

```

procedure CLIP_3D_MOD (  $\mathbf{x}_A$  ,  $\mathbf{x}_B$  );
begin   { for a line    $t_{\min} := -\infty$  ;    $t_{\max} = \infty$  ;   }

     $t_{\min} := 0$ ;    $t_{\max} = 1$ ;    $i := 1$ ;    $j := 0$ 
    {  $\rho_1: F_1(\mathbf{x})=A x_1 + C z_1 + D_1 = 0$     $\rho_2: F_2(\mathbf{x})=B y_2 + C z_2 + D_2 = 0$  }
    for  $k := 1$  to  $N_V$  do   {  $N_V$  number of vertices }
         $Q_k := \text{sign}(F_1(\mathbf{x}_k))$ ;   {  $Q_k$  is a vector of int or char types }
    while ( $i \leq N$ ) and ( $j < 2$ ) do
        begin
            {  $\mathbf{x}_k^i$  means a k-th vertex of the i-th triangle }
            { INDEX(i,k) gives the index of k-th vertex }
            { of the i-th triangle, i.e.  $\mathbf{x}_k^i = \mathbf{x}_{\text{INDEX}(i,k)}$  }
            if  $Q_{\text{INDEX}(i,0)} = Q_{\text{INDEX}(i,1)}$  then
                if  $Q_{\text{INDEX}(i,0)} = Q_{\text{INDEX}(i,2)}$  then goto 1;
                {  $\rho_1$  does intersect the i-th triangle }
            if  $\text{sign}(F_2(\mathbf{x}_{\text{INDEX}(i,0)})) = \text{sign}(F_2(\mathbf{x}_{\text{INDEX}(i,1)}))$  then
                if  $\text{sign}(F_2(\mathbf{x}_{\text{INDEX}(i,0)})) = \text{sign}(F_2(\mathbf{x}_{\text{INDEX}(i,2)}))$ 
                    then goto 1;

            { both planes  $\rho_1$  ,  $\rho_2$  intersect the i-th triangle }
            { detailed test finds a value  $t_{\min}$  and  $t_{\max}$  using }
            { a single step of the CB algorithm }
            { ----- }
            {  $\mathbf{n}_i$  must point out of the given convex polyhedron }
             $\xi := \mathbf{s}_i^T \mathbf{n}_i$ ;    $\mathbf{s}_i := \mathbf{x}_i - \mathbf{x}_A$ ;
            if  $\xi <> 0.0$  then
                begin  $t := \mathbf{s}_i^T \mathbf{n}_i / \xi$ ;
                    if  $\xi > 0.0$  then  $t_{\max} := \min ( t , t_{\max} )$ 
                        else  $t_{\min} := \max ( t , t_{\min} )$ ;
                     $j := j + 1$ ;
                end
            else Special case solution; { line is parallel to a facet }
            { ----- }
1:    $i := i + 1$ ;
        end;   { if  $t_{\min} > t_{\max}$  then no intersection point exists }

        if  $t_{\min} \leq t_{\max}$  then SHOW_LINE( $\mathbf{x}(t_{\min})$ , $\mathbf{x}(t_{\max})$ );
    end { of CLIP_3D_MOD };

```

## Algorithm 2

Algorithm for Line Clipping with  $O(N^{1/2})$  complexity

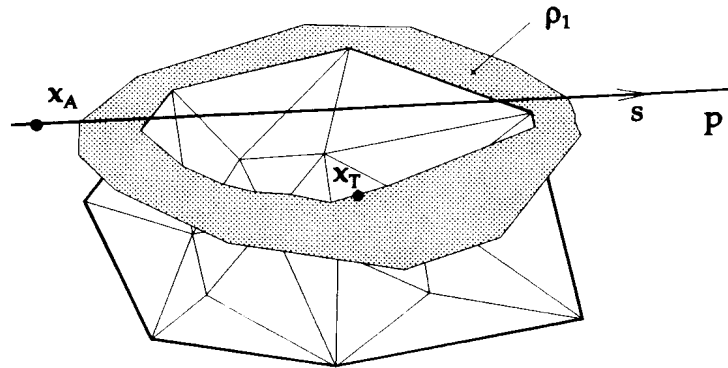
Definition of the plane  $\rho_1$ 

Figure 4

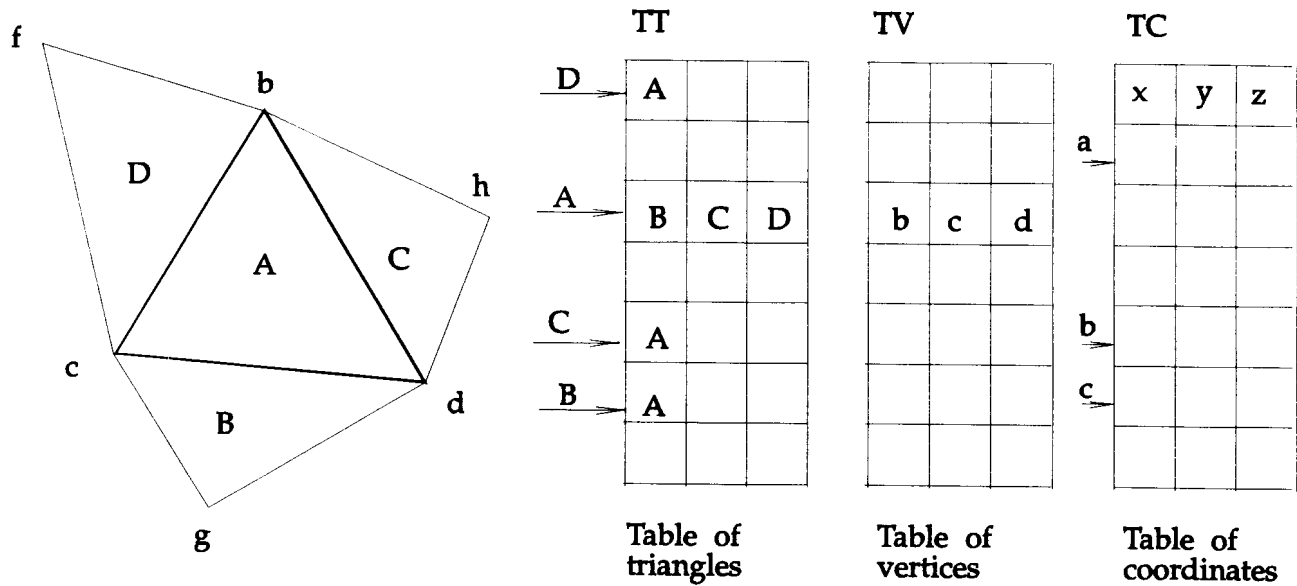
#### 4. Proposed method

To find a facet candidate for the intersection point is a quite complex task and without knowing the "order" of facets an algorithm will be generally of  $O(N)$  complexity. Nevertheless, we can select any triangle (facet)  $\tau_k$  on which the given line does not lie. If we take any point inside of the triangle  $\tau_k$  and the given line we obtain a definition of a plane  $\rho_1$ , see fig.4 on which the line lies. It can be seen that we could develop more efficient strategy for testing triangles if we know facets that are intersected by the plane  $\rho_1$ . Unfortunately the situation is not as simple as in algorithm for line clipping with  $O(\lg N)$  complexity in  $E^2$ , see [SKA94a].

In many applications, data structures that define a convex polyhedron contain information about neighbours of the given triangular facet, see fig.5. It is obvious that we can easily detect which edge of the given triangle  $\tau_k$  is intersected by the plane  $\rho_1$ . In the next step we can take its neighbour which has a common edge with the triangle  $\tau_k$ .

Because of the "knowledge of order" we will test significantly less triangles than  $N$ , but in the worst case we will have to test all  $N$  facets. It means that the algorithm is of  $O(N)$  complexity in the worst case.





Data structure with information about neighbours

Figure 5

Let us consider a surface of a convex polyhedron, e.g. a sphere approximation, that is intersected by a plane  $\rho_1$ . A number of intersected facets by a plane can be estimated as  $N^{1/2}$  in average. This algorithm is briefly described in alg.3. It is necessary to point out that there are some small obstacles in solving singular cases, especially when the plane  $\rho_1$  intersects the triangle in vertex etc.

```

function COMPUTE (  $\mathbf{x}_A$  ,  $\mathbf{x}_B$  ): boolean;
{ computes coefficients  $A_1, B_1, C_1, D_1$  for the plane  $\rho_1$  }
{ separation  $F_1(\mathbf{x}) = A_1x + B_1y + C_1z + D_1$ . }
begin  $\mathbf{x}_T := ( {}^k\mathbf{x}_0 + {}^k\mathbf{x}_1 + {}^k\mathbf{x}_2 ) / 3$ ; { for the k-th triangle }
    Determine coefficients of the plane  $\rho_1$  by solving
    det  $\begin{bmatrix} x & y & z & 1 \\ \mathbf{x}_A & \mathbf{y}_A & \mathbf{z}_A & 1 \\ \mathbf{x}_B & \mathbf{y}_B & \mathbf{z}_B & 1 \\ \mathbf{x}_T & \mathbf{y}_T & \mathbf{z}_T & 1 \end{bmatrix} = 0$ 
    COMPUTE := ( $A_1 \neq 0$ ) or ( $B_1 \neq 0$ ) or ( $C_1 \neq 0$ ) or ( $D_1 \neq 0$ );
end { COMPUTE };
    
```

```

procedure COMPUTE_ORTHOGONAL (  $\mathbf{x}_A$  ,  $\mathbf{x}_B$  );
{ computes coefficients  $A_2, B_2, C_2, D_2$  for the plane  $\rho_2$  }
{  $F_2(\mathbf{x}) = A_2x + B_2y + C_2z + D_2$  }
begin Determine coefficients of the plane  $\rho_2$  by solving

```

$$\det \begin{bmatrix} x & y & z & 1 \\ x_A & y_A & z_A & 1 \\ x_B & y_B & z_B & 1 \\ A_1 & B_1 & C_1 & 1 \end{bmatrix} = 0$$

```

end { COMPUTE_ORTHOGONAL };

```

```

procedure SQRT_CLIP (  $\mathbf{x}_A$  ,  $\mathbf{x}_B$  );

```

```

begin  $t_{\min} := 0$ ;  $t_{\max} := 1$ ;

```

```

    { for a line clipping  $t_{\min} := -\infty$ ;  $t_{\max} := \infty$ ; }

```

```

     $k := N \text{ div } 2$ ; { select a triangle  $\tau_k$  }

```

```

    while not COMPUTE (  $\mathbf{x}_A$  ,  $\mathbf{x}_B$  ,  $k$  ) do

```

```

         $k := (N - 1) * \text{random}$ ;

```

```

        { will be used only if the line  $\mathbf{x}_A\mathbf{x}_B$  lies on the facet }

```

```

    COMPUTE_ORTHOGONAL (  $\mathbf{x}_A$  ,  $\mathbf{x}_B$  );

```

```

    { computes coefficients of the orthogonal plane  $\rho_2$  to  $\rho_1$  }

```

```

     $k0 := -1$ ; { set index of previous triangle }

```

```

    while  $k \neq k0$  do

```

```

        begin { test the triangle  $\tau_k$  against the plane  $\rho_2$  }

```

```

            if TEST (  $\tau_k$  ,  $\rho_2$  ) then

```

```

                begin { a single step of the CB algorithm }

```

```

                    { ----- }

```

```

                    {  $\mathbf{n}_i$  must point out of the polyhedron }

```

```

                     $\xi := \mathbf{s}^T \mathbf{n}_i$ ;  $\mathbf{s}_i := \mathbf{x}_i - \mathbf{x}_A$ ;

```

```

                    if  $\xi <> 0.0$  then

```

```

                        begin  $t := \mathbf{s}_i^T \mathbf{n}_i / \xi$ ;

```

```

                            if  $\xi > 0.0$  then  $t_{\max} := \min ( t , t_{\max} )$ 

```

```

                                else  $t_{\min} := \max ( t , t_{\min} )$ 

```

```

                        end

```

```

                    else Special case;

```

```

                        { line is parallel to a facet }

```

```

                        { ----- }

```

```

                end;

```

```

i := k0;   k0 := k;
k := get_next ( k , i );
{ gives next triangle to  $\tau_k$  that is intersected by }
{ plane  $\rho_1$  and different from previous triangle  $\tau_{k0}$  }
end;
if  $t_{\min} \leq t_{\max}$  then SHOW_3D_LINE (  $x(t_{\min})$  ,  $x(t_{\max})$  );
end { SQRT_CLIP };

```

## Algorithm 3

## 5. Experimental results

The proposed algorithm has been tested against the CB algorithm, see alg.1, and against the algorithm that uses two planes, see alg.2. Data sets of two points that defines lines have been randomly and uniformly generated inside a sphere in order to eliminate an influence of rotation. Convex polyhedrons were generated as N - sided convex polyhedrons that consist of triangular facets and were inscribed into a smaller sphere.

Let us introduce coefficients of efficiency as

$$\nu_1 = \frac{T_{CB}}{T} \qquad \nu_2 = \frac{T_{\rho}}{T}$$

where  $T_{CB}$  ,  $T_{\rho}$  ,  $T$  are execution times in [s] needed by the CB algorithm (CLIP\_3D\_CB), algorithm (CLIP\_3D\_MOD) that uses two planes and the proposed algorithm (SQRT\_CLIP).

Results obtained from experiments are shown in tab.1 and tab.2 for two fundamental cases when any line does not intersect the given polyhedra (0% of intersections) and when all lines intersect (100% of intersections) the given polyhedra and 10 000 clipped lines. All tests were made on PC 486/33 MHz.

N	10	20	50	100	200	500	1000	2000	4000
$T_{CB}$	0.9	2.5	7.8	16.4	33.0	83.5	167.5	335.5	672.1
$T_{\rho}$	0.7	1.2	3.4	6.1	11.4	26.9	52.6	104.7	306.7
T	1.9	2.8	5.8	7.4	11.4	16.9	25.8	36.8	54.9
$\nu_1$	0.5	0.9	1.3	2.2	2.8	4.9	6.5	9.1	12.2
$\nu_2$	0.4	0.4	0.6	0.8	1.0	1.6	2.0	2.9	5.6

Efficiency coefficients  $\nu$  for 0% of intersections

Table 1

N	10	20	50	100	200	500	1000	2000	4000
$T_{CB}$	1.3	3.1	8.1	16.6	33.4	84.5	169.1	338.0	676.5
$T_{\rho}$	4.4	5.4	8.1	10.6	16.2	31.6	54.8	101.0	190.7
$T$	5.4	6.4	9.2	10.1	13.9	18.6	25.8	34.0	49.1
$\nu_1$	0.2	0.5	1.0	1.6	2.4	4.5	6.5	10.0	13.8
$\nu_2$	0.8	0.8	0.9	1.0	1.2	1.7	2.1	2.9	3.9

Efficiency coefficients  $\nu$  for 100% of intersections

Table 2

The proposed SQR\_CLIP algorithm is always faster than the CB algorithm for  $N \geq 50$ , see tab.1 and tab.2, and for  $N \geq 200$  is faster than the algorithm that utilizes planes, see alg.2 and [SKA94b].

## 6. Conclusion

The new efficient algorithm for clipping lines against convex polyhedron was developed with  $O(N)$  complexity in the worst case and expected  $O(N^{1/2})$  complexity. Algorithm does not strictly require triangular facets. The given facets should be oriented. All tests were implemented in C++ on PC 486/33 MHz.

## 7. Acknowledgments

The author would like to express his thanks to students of Computer Graphics courses at the University of West Bohemia in Plzen and Charles's University in Prague who stimulated this work, especially to P.Sebránek and J.Jirák for their careful tests verification and implementation of the proposed algorithm, suggestions and critical comments.

## 8. References

- [CYR79a] Cyrus, M., Beck, J.: Generalized Two and Three Dimensional Clipping, Computers & Graphics, Vol.3, No.1, pp.23-28, 1979.
- [FOL90a] Foley, D.J., van Dam, A., Feiner, S.K., Huges, J.F.: Computer Graphics - Principles and Practice, Addison Wesley, 2nd ed., 1990.

- [SKA93a] Skala, V.: Preprint No.67 Univ. of West Bohemia, Plzeň, 1994. An Efficient Algorithm for Line Clipping by Convex Polygon, Computers & Graphics, Vol. 17, No.4, Pergamon Press, pp.417-421, 1993.
- [SKA94a] Skala, V.:  $O(\lg N)$  Line Clipping Algorithm in  $E^2$ , accepted for publication in Computers & Graphics, Vol.4, Pergamon Press, 1994.
- [SKA94b] Skala, V.: An Efficient Algorithm for Line Clipping by Convex and Non-convex Polyhedrons in  $E^3$ , submitted for publication to Computer Graphics Forum (revised version), 1994.