Computers & Graphics, Vol.20, No.4, pp.523-530, 1996.

Line Clipping in E^2 with O(1) Processing Complexity

Václav Skala[1]
Computer Science Department
University of West Bohemia
Univerzitni 22, Box 314, 306 14 Plzeò - Bory
Czech Republic
skala@kiv.zcu.cz http://yoyo.zcu.cz/~skala

Abstract

A new algorithm for line clipping by convex polygon with $O(1)$ processing complexity is presented. It is based on dual space representation and space subdivision technique. The suggested algorithm also demonstrates that pre-processing can be used in order to speed up solution of some problems in computer graphics applications significantly. Theoretical considerations and experimental results are also presented.

Keywords: Line Clipping, Convex Polygon, Computer Graphics, Algorithm Complexity, Geometric Algorithms, Algorithm Complexity Analysis.

1. Introduction

It is well known that in many applications it is possible to use pre-processing in order to speed up the processing. Pre-processing enables to us to decrease run-time complexity significantly because some parts are stable in their expected usage. Of course it can be used in clipping algorithms only in those cases when the clipping polygon is constant. Line clipping algorithms with many modifications in E^2 have been published so far, see [Ska94a] for known references. The Cyrus-Beck (CB) algorithm is very often used for comparisons as its is very stable. The usual complexity of line clipping algorithms in E^2 is $O(N)$ or $O(\lg N)$, where N is a number of polygon edges, see [Ska94a] for references. The pre-processing complexity must also be taken into account for overall computational complexity and algorithm efficiency considerations.

2. Dual space representation

A line r is usually described by the equation

$$ax + by + c = 0$$

and can be rewritten as

$$y = kx + q \qquad \text{if} \qquad b \neq 0$$

resp.

$$x = my + p \qquad \text{if} \qquad a \neq 0$$

or

$$\frac{x}{p} + \frac{y}{q} = 1 \qquad \text{if} \qquad c \neq 0$$

It means that the line $r \in E^2$ can be represented in dual space representation as a point $D(r) = [k, q] \in D(E^2)$, resp. $D(r) = [m, p] \in D(E^2)$, see Fig.2.1, using an asymmetrical model of dual space representation. There is also a possibility to use the a symmetrical model where the line r is represented as a point $D(r) = [p, q] \in D(E^2)$, but this representation is

[1] Supported by the grant UWB-156/95; Accepted for publication in Computers&Graphics, Vol.20, No.4.,1996

not convenient for our considerations. For more theoretical background of dual space representation, see [Sto89a], [Zac95a]. Dual space representation has very interesting properties and applications can be found in [Nie95a], [Kol94a], [Zac95b]. Generally it is possible to show relations between fundamental geometric primitives by Table 1.1. In the following text we will consider situations in E^2 only. A similar approach can be used for other geometric primitives.

Komentář [VS1]:

Space	Euclidean representation	Dual representation
E^2	line point	point line
E^3	plane line point	point line plane

Fundamental relations between geometric primitives
Table 1.1.

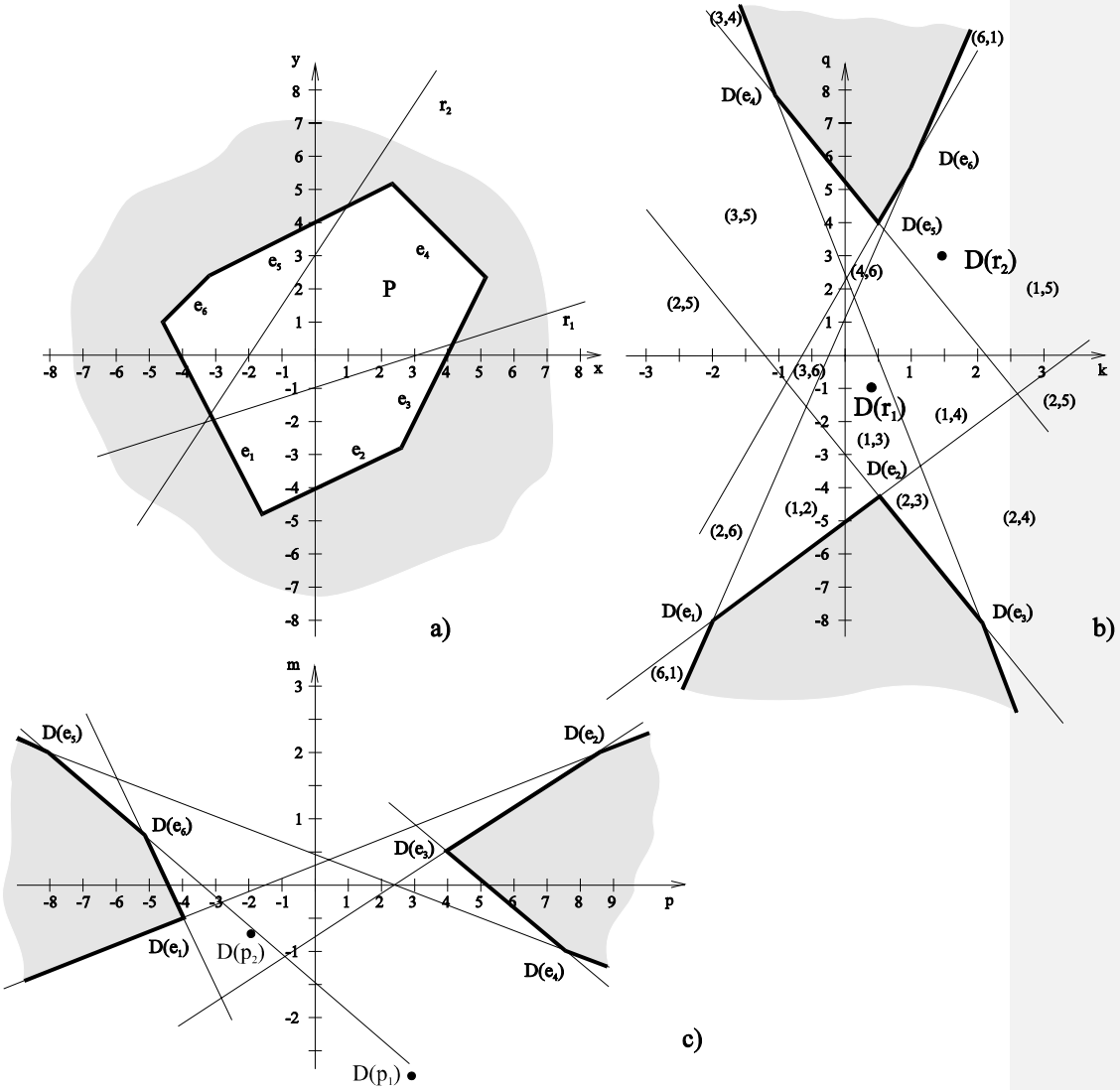
It can be shown that a convex polygon $P \in E^2$, see Fig.2.1.a, can be represented by an infinite area in the dual space $D(E^2)$ if asymmetrical model is used for dual space representation [Zac95a], see Fig.2.1.b, resp. Fig.2.1.c. It can be also shown that the line r intersects the polygon edges (e_a, e_b) **if and only if** the point $D(r)$ lies in a zone (a,b) in dual space representation. The line r_1 intersects edges (e_1, e_2) and therefore the point $D(r_2)$ lies in the zone $(1,2)$, similarly for the line r_2 .

The **detection** whether a line r intersects a convex polygon P is dual to the modified well known Point-in-polygon test. Algorithms for Point-in-polygon test usually have $O(N)$ or $O(\lg N)$ processing complexities. The algorithm complexity can be reduced to an $O(1)$ processing complexity without using parallel processing if pre-processing is used, see [Ska94b] for details.

However the line clipping problem solution generally consists of two steps:

- detection whether the given line intersects the polygon (dual to the Point-in-polygon test),
- selection of polygon edges which are intersected by the given line and computation of intersection points.

It means that the line clipping problem solution is more complex than the Point-in-polygon test. Nevertheless this considerations lead to the line clipping algorithm with $O(\lg_2 N)$ processing complexity described in [Ska94a].



Dual space representation
Figure 2.1

3. Semidual space representation and space subdivision

There are two problems if dual space representation is used that must be solved:
- dual space polygon representation and zones are **infinite** and it is difficult to represent them,
- it is necessary to find a fast method for determining in which zone the point $D(r)$ lies.

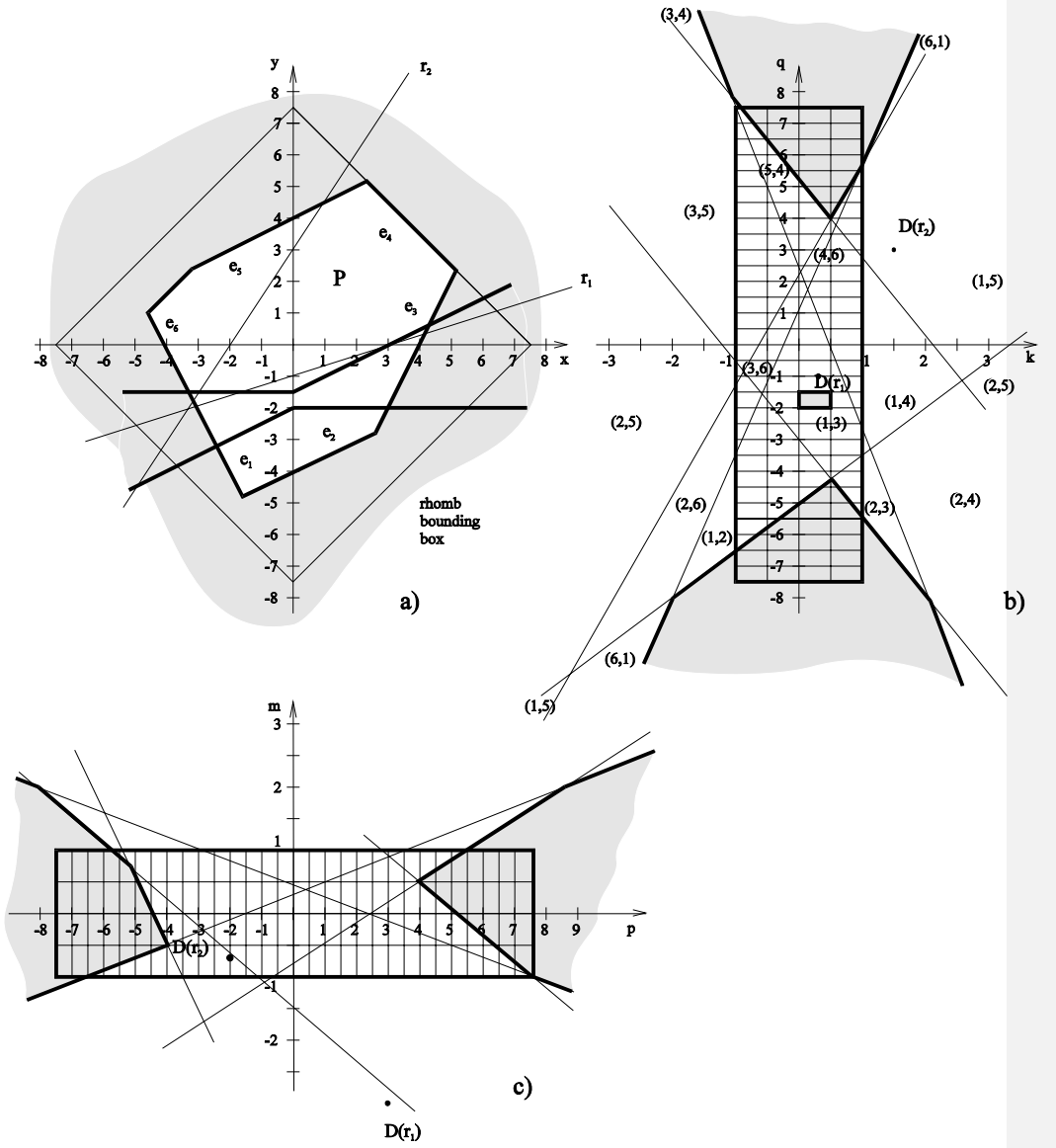
The given line can be represented as

$$y = kx + q \quad \text{if} \quad |k| \le 1$$

or

$$x = my + p \quad \text{if} \quad |m| < 1$$

This means that k, resp. m values are limited.



Semidual space representation
Figure 3.1

Let us consider a rectangular bounding rhomboid so that the given polygon P is inside of that rectangle, Fig.3.1.a. It can also be seen that values q, resp. p are limited. Then values $[k,q]$, resp. $[m,p]$ are from the limited area $<-1,1>x<-a,a>$ in both space representations. We will denote those two limited spaces as **semidual spaces**. It can be seen that those two limited semidual spaces represent the original dual space.

Fast methods for detection of the zone in which zone a point $D(r)$ lies are needed. Several sophisticated techniques have been developed as a part of computational geometry, see [Pre85a]. One possibility is a usage of the space subdivision technique. If semidual spaces for (k,q), resp. (m,p) are subdivided into small rectangles then it is possible to pre-compute a list AEL (Active Edge List) of polygon edges which interfere in semidual space with the given rectangle. If the rectangles are small enough then each list is empty or will contain only two polygon edges.

It is necessary to point out that the rhomboid that bounds the polygon must be as small as possible. Generally the limits for p and q axes can differ and decrease memory requirements.

4. Principle of the proposed algorithm

Let us assume the situations shown in Fig.3.1.a. The proposed algorithm for line clipping with O(1) processing time can be described by the sequence of steps:

Determine whether (k,q) or (m,p) semidual space shall be used for the given line;
if (k,q) semidual space is used { $|\Delta x| \geq |\Delta y|$ }
then
begin compute values $[k,q]$ for the given line r;
find a rectangle that contains the point $D(r)$;
for all members of the AEL list compute intersections with the line r
and test if exist;
end
else similarly for (m,p) semidual space;

Algorithm 4.1

It is obvious that the algorithm complexity does not depend on the number of polygon edges but on the length of the AEL list associated with each rectangle only. If rectangles are small enough then just two edges can be expected in the list nearly for all rectangles. Because all steps in Alg.4.1 have $O(1)$ complexity the whole algorithm has $O(1)$ complexity, too. It is necessary to point out that number of members in AEL list depends on the number of subdivision in (k,q), resp. (m,p) spaces and also on geometric shape of the given polygon, see [Ska94b]. Detailed algorithm is described in Alg.4.2.

```
global      k0 := Nq / (2*a);      k1 := Nk / 2;    { global constants }
            k3 := Np / (2*a);      k4 := Nm / 2;
t0 := +∞;   t1 := -∞;      { initialisation - empty interval  < t0 , t1 > = ∅ }
Δx :=  xB - xA;        Δy := yB - yA;
if | Δx | ≥ | Δy | then { select (k,q) semidual space }
begin  k := Δy/Δx;    q := yB - k*xB;
       i := int (( q + a ) * k0) + 1;
       j := int (( k + 1) * k1) + 1;
       test := false;   { test all members of AEL list for zone (i,j) and compute values t }
                        {for the line intersection points of selected edges }
       test := COMPUTE (ZONE(i,j), t0,t1);  { if intersection exists  then test = true }
end
else
```

```
begin  m := Δx/Δy;    p := xB - m*yB;
       i := int (( p + a ) * k3) + 1;
       j := int (( m + 1) * k4) + 1;
       test := false;   {test all members of AEL list for zone (i,j) and compute values t}
                        {for the line intersection points of selected edges }
       test := COMPUTE (ZONE(i,j), t0,t1);          { test = true  if intersection exists }
end;

if line segment clipping then < t0,t1> := < t0,t1> ∩ <0,1>;
test := test and (< t0 , t1 > ≠ ∅);
if test then  { intersection exists }
begin
       x0:= xA + Δx * t0;      y0:= yA + Δy * t0;
       x1:= xA + Δx * t1;      y1:= yA + Δy * t1;
end;
```

Algorithm 4.2

The function COMPUTE is based on the CB algorithm that is performed only for edges included in the AEL list associated with the selected ZONE(i,j).

It can be shown that computation of AEL lists for all zones is of $O(N * N_k * N_q)$, resp. $O(N * N_m * N_p)$ complexity, where:

- N is a number of edges of the given polygon,
- N_k, resp. N_q is a number of subdivision in the direction of k, resp. q,
- N_m, resp. N_p is a number of subdivision in the direction of m, resp. p.

5. Construction of the AEL list

How to create the AEL lists is a rather tricky problem because if one would like to set up the AEL list directly an algorithm is quite complicated. A simple solution for setting up the AEL lists for all zones in (k,q) semidual space is described by Alg.5.1.

```
for i:=1 to Nk-1 do
       for j:=1 to Nq-1 do
              for k:=1 to N do
                     if edgek interferes with the zone (i,j) defined
                                          by corners (i,j) and (i+1,j+1)
                            then add edgek into the AELij list;
```

Algorithm 5.1

Because N is usually small and there are no special cases this method is fast enough. The AEL list for (m, p) semidual space can be determined in a similar way.

Now it is necessary to find a criterion how to determine the N_k and N_q, resp. N_m and N_p values because we would like to have just one pair of polygon edges for each AEL. It can be shown that for (k,q) semidual space it is necessary to calculate $[k,q]$ values for all polygon edges from the equation.

$$y = kx + q$$

Then

$N_q > 2a / \Delta y$ where $\Delta y = \min \{| y_i - y_j |\}$ for all i, j & $i \neq j$ & $\Delta y > 0$

$N_k > 2 / \Delta k$ where $\Delta k = \min \{| k_i - k_j |\}$ for all i, j & $i \neq j$ & $\Delta k > 0$

and $i, j \in \{1, \ldots, N\}$

Similarly for (m, p) semidual space

$$x = my + p$$

and

$N_p > 2a / \Delta x$ where $\Delta x = \min \{| x_i - x_j |\}$ for all i, j & $i \neq j$ & $\Delta x > 0$

$N_m > 2 / \Delta m$ where $\Delta m = \min \{| m_i - m_j |\}$ for all i, j & $i \neq j$ & $\Delta m > 0$

$i, j \in \{1, \ldots, N\}$

$[x_i, y_i]$ are vertices of the given polygon and k_i is the slope of the i-th edge, similarly for m_i.

It means that the N_k and N_q, resp. N_m and N_p values depend on geometric shape of the given polygon. For detailed description, see [Ska94b].

6. Theoretical considerations and experimental results

The proposed algorithm has been tested and compared with the CB algorithm as the CB algorithm is very stable and its behaviour more or less does not depend on geometric properties of the given polygon and clipped lines. Because the proposed algorithm is supposed to be superior over other modifications of the CB algorithm it is necessary to a make theoretical estimation of its efficiency.

Before making any experiments it is necessary to point out that the time needed for operations (:= , < , ± , * , /) differs from computer to computer, see tab.6.1.

Float	:=	<	±	*	/
Time	33	50	16	20	114

Times for 5.10^7 operations for a PC 486DX/33 MHz

Table 6.1

Let us assume that N is the number of edges of the given polygon. For algorithm efficiency considerations we will consider:

- CB algorithm complexity, see [Ska93a], can be described as

$$T_{CB} = (8,3,6,4,0) + (5,3,7,4,1) * N$$

and time of computation can be estimated

$$T_{CB} = 590 + 621 * N$$

- ECB algorithm with the usage of separation function, see [Ska93a], has complexity

$$T_{ECB} = (15,3,11,14,2) + (3,1,1,3,0) * N$$

and time of computation can be estimated

$$T_{ECB} = 1329 + 257 * N$$

- $O(\lg N)$ algorithm [Ska94a] with complexity defined as

$$T_{\lg} = (14,4,11,15,2) + (2,4,6,6,0) * N$$

and time of computation can be estimated

$$T_{\lg} = 1267 + 376 * \lg(N+1)$$

- proposed $O(1)$ with complexity defined as

$$T_{O(1)} = (22,6,17,19,3)$$

and time of computation can be estimated

$$T_{O(1)} = 2020$$

Let us introduce algorithm efficiency coefficients as:

$$\nu_1 = \frac{T_{CB}}{T_{O(1)}} \qquad \nu_2 = \frac{T_{ECB}}{T_{O(1)}} \qquad \nu_3 = \frac{T_{\lg}}{T_{O(1)}}$$

then the expected efficiency of the proposed algorithm is described by Tab.6.2, Fig.6.1.- 6.2.

N	3	4	5	10	50
CB	2 453	3 074	3 695	6 800	31 640
ECB	2 100	2 357	2 614	3 899	14 179
O(lg N)	2 019	2 395	2 395	2 771	3 523
O(1)	2 020	2 020	2 020	2 020	2 020
ν_1	1,3	1,6	1,9	3,4	15,7
ν_2	1,1	1,2	1,3	2,0	7,1
ν_3	1,0	1,2	1,2	1,4	1,8

Theoretical estimation of time and efficiency
Table 6.2

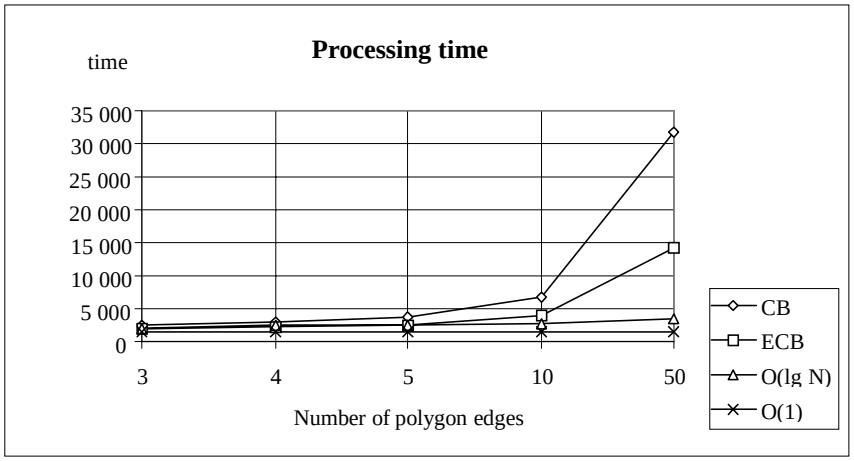


Figure 6.1

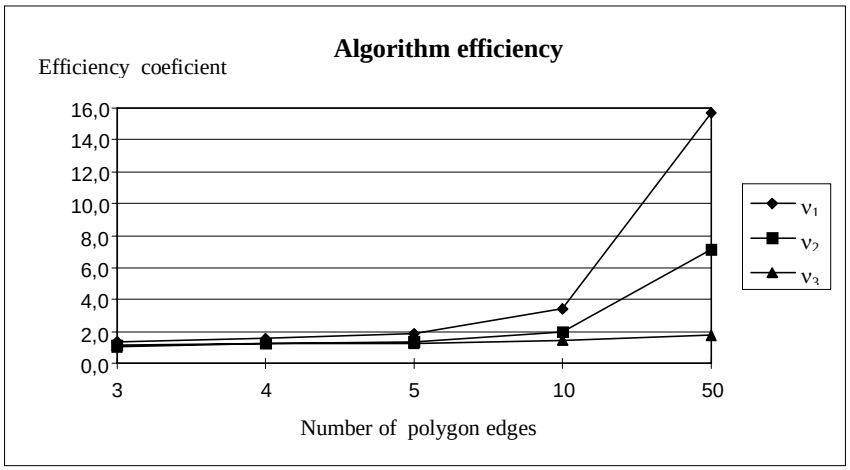


Figure 6.2

The proposed algorithm has been experimentally compared with the CB algorithm and obtained experimental results are shown in Tab.6.3 and Fig.6.3.The difference between theoretical estimation and experimental results for processing time, see Tab.6.3, is very small and proved the theoretical analysis of the proposed algorithm.

N	3	4	5	10	50
ν_1	1,1	1,5	1,7	3,5	15,3

Experimental results without pre-processing
Table 6.3

N	3	4	5	10	50
ν_1	1,1	1,2	1,4	1,8	2,6

Experimental results for 10000 lines including pre-processing
Table 6.4

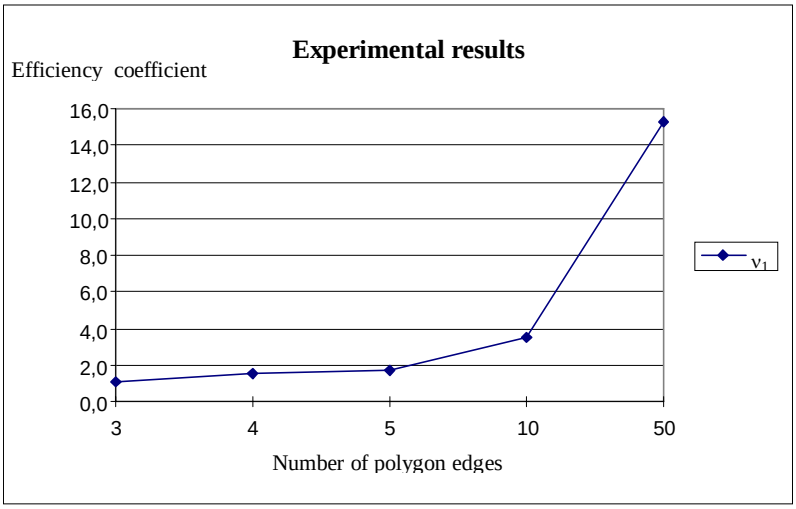


Figure 6.3

For partial comparative study with other approaches see [Ska95a]. Full comparison wit Cyrus-Beck algorithm can be found in [Ska95c].

7. Conclusion

A new line clipping algorithm against a convex polygon in E^2 was developed. It is based on dual space representation and space subdivision technique. The proposed algorithm is convenient for those applications where the clipping area is stable and many lines are clipped. The algorithm is $O(1)$ processing complexity and faster than the CB, ECB and O(lg N) algorithms in the anticipated applications.

The presented approach can be applied in many areas of computer graphics and there is a hope that it can be used to speed up line clipping algorithms in E^3 and ray tracing techniques.

8. Acknowledgements

The author would like to express his thanks to Mr.P.Lederbuch for careful implementation and testing algorithms and to all who contributed to this work, especially to recent PhD. student Ms.I.Kolingerová, MSc. and PhD. students of Computer Graphics courses at the University of West Bohemia in Plzeò who stimulated this work and proposed many suggestions.

9. References

[Kol94a] Kolingerová,I.: Dual Representation and Its Usage in Computer Graphics, PhD Thesis (in Czech), Univ. of West Bohemia, Plzeò, 1994.

[Nie95a] Nielsen,H.P.: Line Clipping Using Semi-Homogeneous Coordinates, Computer Graphics Forum, Vol.14, No.1, pp.3-16, 1995.

[Pre85a] Preparata,F.P., Shamos,M.I.: Computational Geometry: An Introduction, Springer Verlag, 1985.

[Ska93a] Skala,V.: 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, Computers & Graphics, Vol.18, No.4, Pergamon Press, pp.517-524, 1994.

[Ska94b] Skala,V.: Point-in-Polygon with O(1) Complexity, TR 68/94, Univ. of West Bohemia, Plzeò, 1994.

[Ska95a] Skala,V., Kolingerová,I., Bláha,P.: A Comparison of 2D Line Clipping Algorithms, Machine Graphics and Vision, Vol.3, No.4, pp. 625-633, 1995.

[Ska95b] Skala,V., Lederbuch,P.: A Comparison of O(1) and Cyrus-Beck Line Clipping Algorithm in E^2, submitted SSCG96 Int.Conf., Slovak Republic, 1995.

[Sto89a] Stolfi,J.: Primitives for Computational Geometry, Report 36, SRC DEC System Research Center, 1989.

[Zac95a] Zachariáš,S.: Duality and Complexity (in Czech), TR 81/95, Univ.of West Bohemia, Plzeò, 1995.

[Zac95b] Zachariáš,S.: Projection in Barycentric Coordinates, submitted to WSCG96 Int.Conf., Univ.of West Bohemia, Plzeò, 1995.