A COMPARISON OF A NEW O(1) AND THE CYRUS-BECK LINE CLIPPING ALGORITHMS IN E^2

Václav SKALA, Pavel LEDERBUCH[1]
Department of Information Technology and Computer Science
University of West Bohemia
Univerzitní 22, Box 314
306 14 Plzen
CZECH REPUBLIC
{skala, lederbuc}@kiv.zcu.cz

ABSTRACT

A comparison of a new algorithm for line clipping in E^2 for convex polygons with O(1) processing complexity and the Cyrus-Beck algorithm is presented. The algorithm is based on the dual space representation and a space subdivision technique. The algorithm demonstrates that preprocessing can be used to speed up line clipping significantly. Theoretical analysis and detailed experimental results are also presented.

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

INTRODUCTION

Many applications in computer graphics use line clipping algorithms. If the given polygon is fixed, preprocessing can speed up the processing time. There are many well-known modified line clipping algorithms in E^2 (see for example Skala94 for more references). The usual complexities of line clipping algorithms in E^2 are O(N) or O(lg N). The O(N) Cyrus-Beck (CB) algorithm is a very often used benchmark as it is very stable [Cyrus-Beck79].

THE SEMIDUAL SPACE REPRESENTATION

The standard representation of a line is given by Eq. 1

$$ax + by + c = 0 \qquad (1)$$

That can be rewritten as

$$\text{if} \quad |k| \le 1 \qquad y = kx + q \qquad (2)$$

and

$$\text{if} \quad |m| < 1 \qquad x = my + p \qquad (3)$$

respectively.

These equations show that a line $r \in E^2$ can be represented as a point $D(r) = [k, q] \in D(E^2)$, respectively $D(r) = [m, p] \in D(E^2)$. This representation will be called „dual space representation“. This representation possesses very interesting properties and applications that can be found in Stolfi89, Kolingerová94, Nielsen95, Zachariáš95.

[1] Supported by the grant UWB-156/1995-96

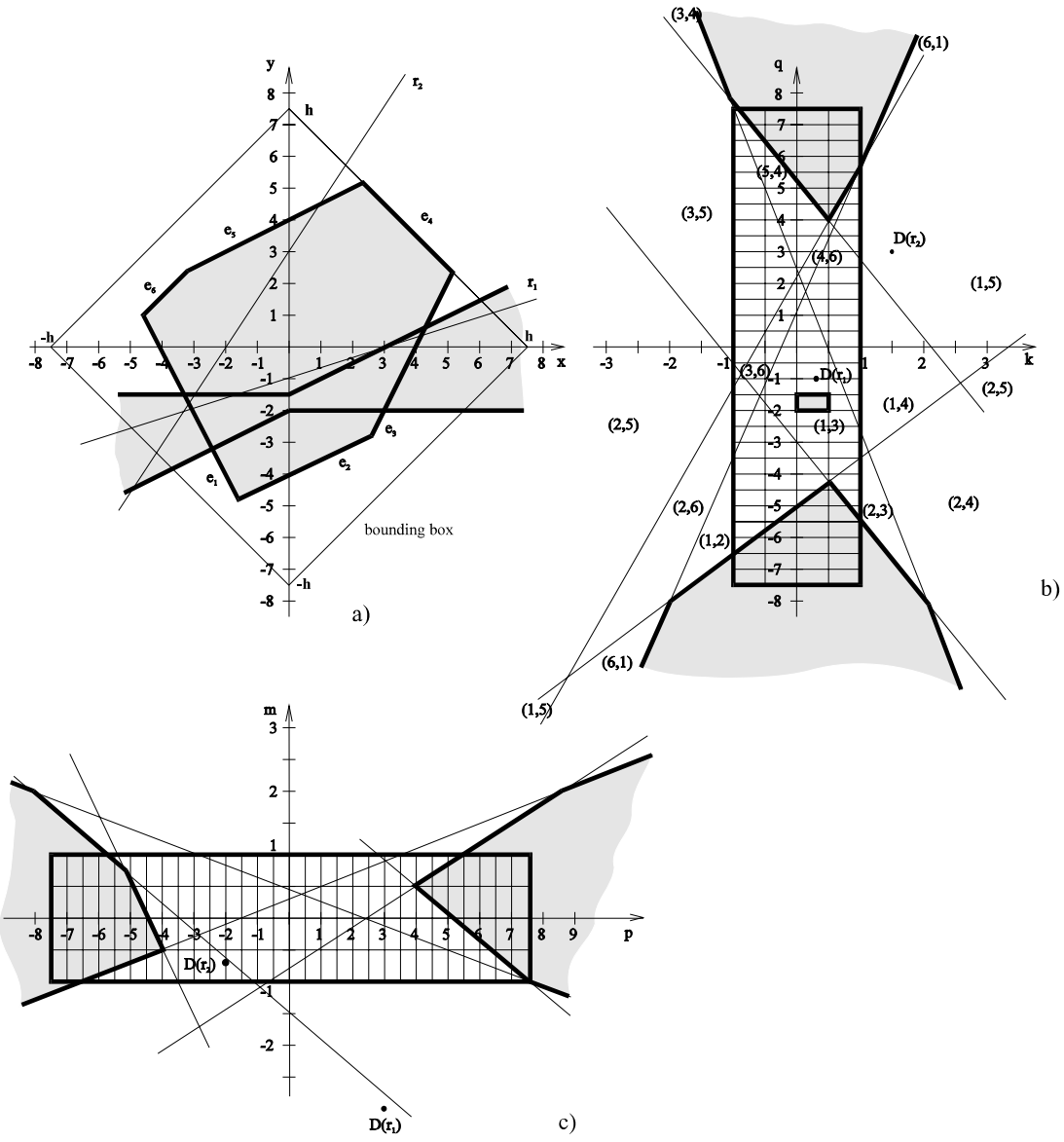

Fig. 1. The semidual space representation.

The values *k* and *m* are limited to <-1, 1> (see above equations). Let us consider a bounding box so that the given polygon is inside a rectangle. Then values of *k*, *q* and *m*, *p* are from the bounding box $< -1, 1 >, < -h, h >$ in both space representations (see Fig. 1). These two finite spaces are named as semidual spaces.

SPACE SUBDIVISION

A space subdivision technique is used to detect the region on which a point *D(r)* lies. The semidual spaces for *(k, q)*, respectively *(m, p)* are subdivided into small rectangles. Each rectangle is a dual representation of a region in E^2 (see Fig. 1a). For each region in E^2 it is possible to compute a list of polygon edges that intersects it. Such list is named the Active Edge List (AEL).

It is necessary to point out that the number of elements in AEL depends on the geometric shape of the given polygon and also on the fineness of the subdivision in *(k, q)*, respectively *(m, p)* spaces. If the rectangles are small enough (i.e., the subdivision is fine) then each list contains no more than two edges of the given polygon.

It is necessary to find a criterion for the subdivision of semidual spaces so that just one pair of polygon edges is in the AEL.

For the *(k, q)* semidual space the Eq. 4 is used

$$y = kx + q \tag{4}$$

Then

$$n_q > 2a / \Delta y \tag{5}$$

where $\Delta y = \min \{| y_i - y_j |\}$

$$i, j \in \langle 0, n \rangle \ \& \ i \neq j \ \& \ \Delta y > 0$$

$$n_k > 2 / \Delta k \tag{6}$$

where $\Delta k = \min \{| k_i - k_j |\}$

$$i, j \in \langle 0, n \rangle \ \& \ i \neq j \ \& \ \Delta k > 0$$

Similarly for the (m, p) semidual space

$$x = my + p \tag{7}$$

and

$$n_p > 2a / \Delta x \tag{8}$$

where $\Delta x = \min \{| x_i - x_j |\}$

for all $i, j \ \& \ i \neq j \ \& \ \Delta x > 0$

$$n_m > 2 / \Delta m \tag{9}$$

where $\Delta m = \min \{| m_i - m_j |\}$

for all $i, j \ \& \ i \neq j \ \& \ \Delta m > 0$

The interpretation is that the n_k and n_q, respectively n_m and n_p, values depend on the geometric shape of the given polygon.

The above conditions guarantee that each list of AEL contains up to three edges. It is necessary to point out that these conditions may extremely increase the subdivisions of semidual spaces and the memory requirements may be above the system possibilities. That is why these conditions may not be realized. It is then essential to find an optimal level of subdivision which will subdivide the semidual spaces sufficiently, but, on the other hand, will do not exceed the available memory. Experimental results of the space subdivision for a polygon with N = 10 are presented in Table 1, Table 2, Fig. 2, and Fig. 3.

n_k \ n_q	1	2	10	20	100	200	1000	2000	10000
1	10,00	8,50	4,75	4,03	3,27	3,18	3,12	3,11	3,10
10	10,00	6,20	3,02	2,50	1,81	1,72	1,65		
100	10,00	6,02	2,63	2,17	1,77				
1000	10,00	6,00	2,60						

Table 1. Number of edges in the AEL (N = 10)

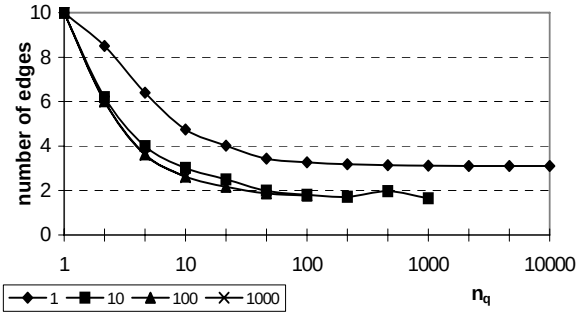

Fig. 2. Number of edges in the AEL dependent on subdivision in the direction q
(N = 10, n_k is 1, 10, 100 and 1000)

n_q \ n_k	1	2	10	20	100	200	1000	2000
1	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
10	4.75	3.93	3.02	2.79	2.63	2.61	2.60	
100	3.27	2.27	1.81	1.81	1.77			
1000	3.12	2.10	1.65					

Table 2. Number of edges in the AEL (N = 10)

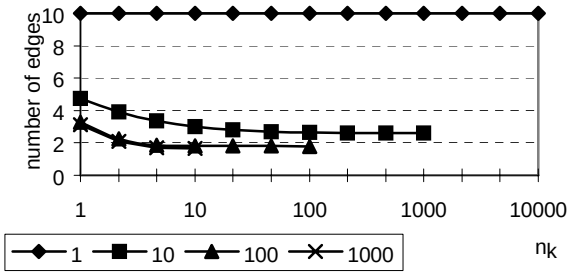

Fig. 3. Number of edges in the AEL dependent on subdivision in the direction k
(N = 10, n_q is 1, 10, 100 and 1000)

The experimental results show that subdivision in the direction q, respectively p, is more significant than subdivision in the direction k, respectively m. Fig. 2 and Fig. 3 show that the adequate numbers of subdivision steps are $n_q = 10 * N$ and $n_k = N$. The test has been made for polygons with N = 3, 4, 5, 10, 20, 50, 100 with the same results.

The construction time of the AEL is presented in Table 3 and Fig. 4.

n_q \ N	3	5	10	20	50
1	0,00	0,00	0,06	0,10	0,22
2	0,06	0,05	0,05	0,11	0,28
5	0,05	0,06	0,11	0,28	0,55
10	0,11	0,11	0,28	0,49	1,15
20	0,17	0,27	0,44	0,87	2,20
50	0,44	0,66	1,20	2,25	5,39
100	0,88	1,32	2,36	4,45	10,76
200	1,75	2,59	4,67	8,90	21,48
500	4,40	6,49	11,70	22,14	53,61
1000	8,90	13,24	23,95	45,48	110,07

Table 3. Preprocessing time of the AEL (subdivision in the direction k for $n_k = 10$)

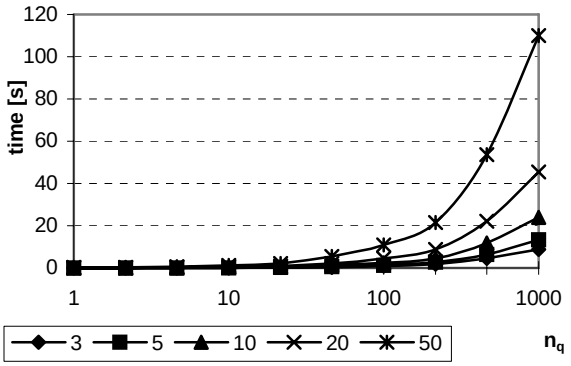

Fig. 4. Preprocessing time of the AEL (subdivision in the direction k for $n_k = 10$)

THE O(1) LINE CLIPPING ALGORITHM

Assuming the conditions of Fig. 1, the algorithm for line clipping with an O(1) processing time complexity can be described by as follows:

Algorithm 1

Determine whether the (k,q) or (m,p) semidual spaces shall be used for the given line;

if $|\Delta x| \geq |\Delta y|$ **then**
begin
compute values $[k,q]$ for the given line r;
find a rectangle containing the point $D(r)$;
for all members of the AEL compute the intersections with the line r and test if they exist;
end
else similarly for the (m,p) semidual space;

Because all steps in Alg. 1 have an O(1) complexity, the algorithm on the whole also has an O(1) complexity. The detailed algorithm is described below.

Algorithm 2

```
global { global constants }
k0 := nq / (2*a); k1 := nk / 2;
k3 := np / (2*a); k4 := nm / 2;
t0 := +∞; t1 := -∞;
{ initialisation - interval  < t0 , t1 > = ∅ }
Δx :=  xB - xA;  Δy := yB - yA;
if | Δx | ≥ | Δy | then { (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 the AEL for region(i,j)}
   { compute the value of t for the line }
   { intersection points of selected edges }
   test := COMPUTE (REGION(i,j), t0,t1);
   { if an 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 the AEL for region(i,j)}
   { compute the value of t for the line }
   { intersection  points of selected edges }
   test := COMPUTE (REGION(i,j), t0,t1);
   { test = true  if an intersection exists }
end;
if line segment clipping
   then < t0,t1> := < t0,t1> ∩ <0,1>;
test := test and (< t0 , t1 > ≠ ∅);
if test then  { an intersection exists }
begin
   x0:= xA + Δx * t0;   y0:= yA + Δy * t0;
   x1:= xA + Δx * t1;   y1:= yA + Δy * t1;
end;
```

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

It can be shown that computation of the AEL's for all regions is of $O(N*n_k*n_q)$, respectively $O(N*n_m*n_p)$ complexity, where:

- N is the number of edges of the given polygon,
- n_k , respectively n_q are the number of subdivisions in the direction k, respectively q,
- n_m , respectively n_p are the number of subdivisions in the direction m, respectively p.

THEORETICAL CONSIDERATIONS OF THE O(1) ALGORITHM

The present O(1) algorithm has been tested and compared with the Cyrus-Beck algorithm. The reason for the choice of the CB algorithm is that this algorithm is numerically very stable and its behaviour does not depend on the geometric properties of the given polygon and clipped lines.

Before carrying out any experiments, it is necessary to consider that the processing time of operations such as $:=, <, \pm, *, /$ may differ significantly from computer to computer (as shown in Table 4).

Float	:=	<	±	*	/
Time [s]	33	50	16	20	114

Table 4. Times for 5.10^7 operations for a PC 486DX/50 MHz

Let us assume that N is the number of edges of the given polygon. The CB algorithm complexity can be expressed as

$$T_{CB} = (8,3,6,4,0) + (5,3,7,4,1) * N \qquad (10)$$

and the computation time can be estimated by

$$T_{CB} = 590 + 621 * N \qquad (11)$$

The theoretical complexity of the O(1) algorithm can be expressed as

$$T_{O(1)} = (22,6,17,19,3) \qquad (12)$$

and the computation time can be estimated by

$$T_{O(1)} = 2020 \quad \text{(constant time)} \qquad (13)$$

Let it be defined the algorithm efficiency coefficients as

$$\nu_1 = \frac{T_{CB}}{T_{O(1)}} \qquad (14)$$

$$\nu_2 = \frac{T_{CB}}{T_{O(1)} + T_{prep}} \qquad (15)$$

then the expected efficiency of the O(1) algorithm is shown in Table 5.

N	3	4	5	10	50
ν_1	1.3	1.6	1.9	3.4	15.7

Table 5. Theoretical estimation of the O(1) algorithm efficiency.

RESULTS OF THE COMPARISON BETWEEN THE O(1) ALGORITHM AND THE CB ALGORITHM

Experimental results for the processing time in seconds are presented in Table 6, Table 7 and Fig. 5.

N	3	4	5	10	20	50
T_{CB}	0,99	1,26	1,59	3,13	6,59	15,38
T_{prep}	0,44	0,55	0,66	1,20	2,25	5,39
$T_{O(1)}$	0,50	0,50	0,50	0,54	0,55	0,55
$T_{prep}+T_{O(1)}$	0,94	1,05	1,16	1,74	2,80	5,94
ν_1	2,0	2,5	3,2	5,8	12,0	28,0
ν_2	1,1	1,2	1,4	1,8	2,4	2,6

Table 6. Experimental results for processing times
(M=10.000, P=0%, coefficients of O(1) are n_k=10, n_q=50)

N	3	4	5	10	20	50
T_{CB}	0,99	1,32	1,65	3,13	6,15	15,32
T_{prep}	0,44	0,55	0,66	1,20	2,25	5,39
$T_{O(1)}$	1,32	1,32	1,32	1,32	1,32	1,49
$T_{prep}+T_{O(1)}$	1,76	1,87	1,98	2,52	3,57	6,88
ν_1	0,8	1,0	1,3	2,4	4,7	10,3
ν_2	0,6	0,7	0,8	1,2	1,7	2,2

Table 7. Experimental results for processing times
(M=10.000, P=100%, coefficients of O(1) are n_k=10, n_q=50)

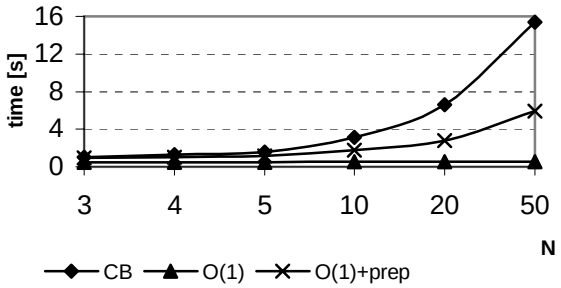

Fig. 5. Experimental results for processing times
(related to the values in Table 6)

A comparison between theoretical estimations and experimental results is presented in Table 8.

N	3	4	5	10	50
ν_1 (theoret.)	1,3	1,6	1,9	3,4	15,7
ν_1 (exp. P=0)	2,0	2,5	3,2	5,8	28,0
ν_1 (exp. P=100)	0,8	1,0	1,3	2,4	10,3

Table 8. Theoretical and experimental efficiencies

Dependence of the processing time of the O(1) algorithm on the probability, that clipped lines intersect the given polygon, is shown in Table 9 and Fig. 6 (for N=10). The experimental results show that the O(1) algorithm is faster than the CB algorithm if the number of clipped lines is greater than 5000. This limit is the same for all probabilities of intersection of a polygon by the clipped line.

M	1000	5000	10000	20000	50000	1E+05
T_{CB}	0,28	1,48	3,02	5,99	15,05	30,10
T_{prep}	1,20	1,20	1,20	1,20	1,20	1,20
$T_{O(1)}$	0,06	0,27	0,50	1,04	2,58	5,16
$T_{O(1)}+T_{prep}$	1,26	1,47	1,70	2,24	3,78	6,36
ν_1	4,7	5,5	6,0	5,8	5,8	5,8
ν_2	0,2	1,0	1,8	2,7	4,0	4,7

Table 9. Processing times for various numbers of clipped lines
(N = 10, coefficients of O(1) are n_k=10, n_q=50)

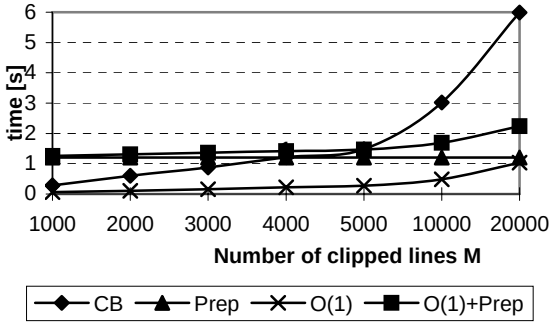

Fig. 6. Processing times for various numbers of clipped lines

The processing time of the proposed O(1) algorithm depends on the probability, that the clipped line intersects the given polygon. It can be observed in Table 10 and Fig. 7. The time complexity of the CB algorithm is nearly constant. The processing time of the O(1) algorithm grows, when the clipped line intersects the polygon. This is caused by the processing time of the test whether a line intersects a polygon. This test is faster for a line which does not intersect the given polygon, than for a line which intersects it.

P	0	10	30	50	70	90	100
T_{CB}	3,0	3,1	3,1	3,1	3,1	3,1	3,2
T_{prep}	1,2	1,2	1,2	1,2	1,2	1,2	1,2
$T_{O(1)}$	0,5	0,6	0,7	0,9	1,1	1,2	1,3
$T_{O(1)}+T_{prep}$	1,7	1,8	1,9	2,1	2,3	2,4	2,5
n_1	1,7	1,7	1,6	1,5	1,4	1,3	1,3
n_2	5,6	5,1	4,3	3,5	2,9	2,6	2,4

Table 10. Processing times for various probabilities of intersection
(N = 10, M = 10.000, coefficients of O(1) are n_k=10, n_q=50)

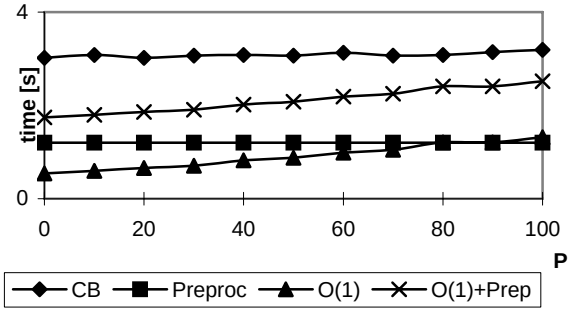

Fig. 7. Processing times for various probabilities of intersection.

CONCLUSION

An algorithm for line clipping of a convex polygon in E^2 with an O(1) processing complexity has been devised and tested. It has been compared with the Cyrus-Beck line clipping algorithm.

The experimental results obtained confirmed the theoretical analysis of the algorithm and its superiority to the CB algorithm. Current experiments are being carried out to evaluate whether a similar approach as the O(1) line clipping algorithm can be used with advantage to speed up line clipping algorithms in E^3 and possibly ray tracing techniques.

All tests were performed by means of programmes implemented in C++ on a 486/50 MHz PC. Detailed reports can be found at the http://herakles.zcu.cz site in PostScript format.

ACKNOWLEDGEMENTS

The authors would like to express their thanks to Miss I. Kolingerová and Mr. B. Sup for their critical comments, and to the students of Computer Graphics at the University of West Bohemia in Plzen for the suggestions that stimulated this project .

REFERENCES

Cyrus, M., Beck, J. 1979 Generalized two and three dimensional line clipping, *Computers & Graphics*, Vol. 3, No. 1, pp 23-28.

Kolingerová, I. 1994 *Dual Representation and its Usage in Computer Graphics, University of West Bohemia,* Plzeò, Czech Republic, PhD Thesis (in Czech).

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

Skala, V. 1994 O(lg N) Line Clipping Algorithm in E^2, *Computers & Graphics*, Vol.18, No.4, pp 517-524.

Stolfi, J. 1989 *Primitives for Computational Geometry, SRC DEC System Research Center*, Research Report 36, January 27 (PhD dissertation).

Zachariáš, S. 1995 *Duality and Complexity (in Czech), University of West Bohemia*, Plzeò, Czech Republic, TR 81/95.

APPENDIX A: SYMBOLS

E^2	Euclidean space
$D(E^2)$	dual representation of an Euclidean space
x, y	point coordinates in E^2
a, b, c	line coefficients in E^2
k, q, m, p	line coefficients in E^2, point coordinates in a semidual space
N	number of edges
M	number of clipped lines
P	probability of intersection of a polygon by a clipped line
r	line to be clipped
n_k, n_q, n_m, n_p	number of subdivision steps in the specified directions
T_{CB}	processing time of the CB algorithm
$T_{O(1)}$	processing time of the O(1) algorithm
T_{prep}	preprocessing time of the O(1) algorithm
ν_1, ν_2	efficiency coefficients