INFORMATICA, 2023, Vol. 34, No. 1, 169–198

DOI: https://doi.org/10.15388/23-INFOR508

A Brief Survey of Clipping and Intersection Algorithms with a List of References (including Triangle-Triangle Intersections)☆

Vaclav SKALA
University of West Bohemia, Faculty of Applied Sciences,
Department of Computer Science and Engineering, Pilsen, CZ 301 00, Czech Republic
e-mail: skala@kiv.zcu.cz

Received: June 2022; accepted: January 2023

Abstract. This contribution presents a brief survey of clipping and intersection algorithms in E^2 and E^3 with a nearly complete list of relevant references. Some algorithms use the projective extension of the Euclidean space and *vector-vector* operations, which support GPU and SSE use.

This survey is intended to help researchers, students, and practitioners dealing with intersection and clipping algorithms.

Key words: intersection algorithms, line clipping, line segment clipping, polygon clipping, triangle-triangle intersection, homogeneous coordinates, projective space, duality, computer graphics, geometry, convex polygon, convex polyhedron.

1. Introduction

Intersection algorithms are key algorithms in many areas, e.g. in geometry intersection algorithms of two lines in E^2 or three planes in E^3, CAD/CAM systems, etc. Many of those algorithms are part of standard courses and based on formulations in the Euclidean geometry, e.g. Schneider and Eberly (2003). However, there is a problem with results in infinity or close to infinity. Some of those can be solved using the projective extension of the Euclidean space and the principle of duality (Johnson, 1996; Skala, 2010). The projective extension of the Euclidean space enables representation of points in infinity and the application of the principle of duality to solve dual problems by the same algorithm (Coxeter and Beck, 1992; Johnson, 1996; Skala, 2008b). Such approach leads to formulations using *vector-vector* operations, which is convenient for GPU and SSE instructions.

Algorithms for intersection computation of different geometric entities in E^2 and E^3 are studied for a long time from various aspects. Their robustness and precision of numerical calculations is severely influenced by the limited numerical accuracy available on today's computer system. It is well known that $(1/3) * 3 \neq 1$ in “the computer world”.

☆The research was supported by the University of West Bohemia – Institutional research support.

Even a simple summation $S = \sum_{i=1}^{n} a_i$ is not easy in the case of large-range data (Skala, 2013b).

It should be noted that, not only in geometry oriented algorithms, a special care has to be devoted to the cases where differences between mathematics with infinite precision and mathematics with a limited precision might cause problems leading to the unexpected and incorrect results, sometimes also leading to disasters.

Unfortunately, programmers and computer scientists are mostly targeted at "the technology of implementation". They have a limited understanding of numerical aspects of today's numerical data representation, limited more or less to the IEEE-754 floating-point representation (Wikipedia, 2021b). Despite the technological progress, the binary128 and binary256 precision are not supported in hardware. It appears that there is no possibility to represent rational, irrational and transcendental numbers used in mathematics, where unlimited accuracy is expected, e.g. what is the difference between the value of π^{π} and $(\textbf{long real}\ pi)^{(\textbf{long real}\ pi)}$ if the IEEE-754 representation is used?

Line, half-line (ray), line segment and triangle-triangle intersection algorithms are considered fundamental in nearly all algorithms dealing with geometrical aspects (Skala, 2022).

2. Projective Space and Principle of Duality

The majority of intersection algorithms have been developed for the Euclidean space representation in spite of the fact that geometric transformations, i.e. projection, translation, rotation, scaling and Window-Viewport etc., use homogeneous coordinates, i.e. projective representation. This results into the necessity to convert the results of the geometric transformations to the Euclidean space using division operation.

2.1. *Projective Extension of the Euclidean Space*

The conversion of a point $\mathbf{x} = [x, y : w]^T$ from the homogeneous coordinates to the Euclidean representation $\mathbf{X} = (X, Y)$ is given as:

$$X = x/w, \qquad Y = y/w \quad \& \quad w \neq 0, \tag{1}$$

where w is the homogeneous coordinate.[1]

It means that a point $\mathbf{X} \in E^2$ is represented by a line in the projective space $[x, y : w]^T$ without the origin, which represents a point in infinity, see Fig. 1.

The extension to the E^3 case is straightforward (Foley *et al.*, 1990).

$$X = x/w, \qquad Y = y/w, \qquad Z = z/w \quad \& \quad w \neq 0, \tag{2}$$

where $\mathbf{x} = [x, y, z : w]^T$.

[1]In mathematics, a different notation $\mathbf{x} = [x_0 : x_1, \dots, x_n]^T$ is used; where x_0 represents the homogeneous coordinate w.

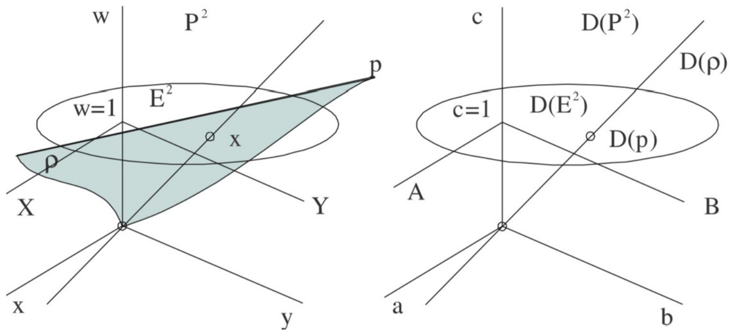

Fig. 1. Projective space and its dual.

The use of the projective extension of the Euclidean space is convenient not only for geometric transformations, as it replaces addition by multiplication in the case of translation operation, but it enables to represent a point in infinity. Also, it enables to express some geometric entities in a more compact form, e.g. a line in the E^2 case as:

$$aX + bY + c = 0, \qquad ax + by + cw = 0, \qquad \mathbf{a}^T\mathbf{x} = 0, \tag{3}$$

where $\mathbf{a} = [a, b : c]^T$, and $\mathbf{x} = [x, y : w]^T$.

It is necessary to note that (a, b) represents the normal vector[2] of a line, while c is related to the distance of a line from the origin of the Euclidean coordinate system. Similarly, a plane in the E^3 case is defined as:

$$aX + bY + cZ + d = 0, \qquad ax + by + cz + dw = 0, \qquad \mathbf{a}^T\mathbf{x} = 0, \tag{4}$$

where $\mathbf{a} = [a, b, c : d]^T$ and $\mathbf{x} = [x, y, z : w]^T$. However, it is necessary to distinguish vectors, as "movable" entities, from "frames", which have the origin as the reference point. It is necessary to note that metric is not defined in the projective space.

In many cases, the principle of duality can be used to derive a solution of a dual problem and have only one programming sequence for both problems, i.e. the primary one and the dual. Figure 1 presents the duality in E^2 – the line p is represented as a point $D(p)$ in the dual space (Stolfi, 1991). Unfortunately, the principle of duality is not usually part of the standard computer science curricula.

2.2. *Principle of Duality*

The principle of duality is one of essential principles in mathematics. In our case of geometric problems described by linear equations, see Eq. (3) and Eq. (4), the principle of duality states that any theorem remains true when we interchange the words:

- "point" and "line" in the E^2 case, resp. "point" and "plane" in the E^3 case,

[2] Actually, it is a bivector (Vince, 2008).

- "lie on" and "pass through", "join" and "intersection" and so on.

Once the theorem has been established, the dual theorem is obtained as described (Johnson, 1996).

In other words, the principle of duality in the E^2 case says that in all theorems it is possible to substitute the term "point" by the term "line" and term "line" by the term "point" and the given theorem remains valid. This helps a lot in the solution of some geometrical problems, similarly in the E^3 case. It means that the intersection computation of two lines is dual to the computation of a line given by two points in the E^2 case.

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} -c_1 \\ -c_2 \end{bmatrix}, \quad \text{i.e. } \mathbf{Ax} = \mathbf{b}, \quad \begin{bmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \text{i.e. } \mathbf{Ax} = \mathbf{0}. \tag{5}$$

It is strange as the usual solution in the first case leads to formulation $\mathbf{Ax} = \mathbf{b}$, while in the second case, the parameters of a line are determined as $\mathbf{Ax} = \mathbf{0}$. However, if the projective representation is used, both cases are solved as $\mathbf{Ax} = \mathbf{0}$ (Skala, 2008b). Similarly, the intersection computation of three planes is dual to the computation of a plane given by three points in the E^3 case.

Generally, a system of linear equations $\mathbf{Ax} = \mathbf{0}$ can be solved as:

$$\mathbf{x} = \mathbf{a}_1 \wedge \mathbf{a}_2 \wedge \cdots \wedge \mathbf{a}_n, \tag{6}$$

where $\mathbf{a}_i$ are rows of the matrix $\mathbf{A}$, $\wedge$ is the outer product, i.e. extended cross product, and $\mathbf{x} = [x_1, \ldots, x_n : w]^T$ is the solution in the homogeneous coordinates. It means that a line given by two points $\mathbf{x}_A$, $\mathbf{x}_B$, resp. an intersection of two lines $\mathbf{p}_1$, $\mathbf{p}_2$ is given in E^2 as $\mathbf{p} = \mathbf{x}_A \wedge \mathbf{x}_B$, resp. $\mathbf{x} = \mathbf{p}_1 \wedge \mathbf{p}_2$, due to the principle of duality.

It should be noted that a line in E^2 can be expressed as:

$$\begin{aligned} &aX + bY + c = 0 \quad \text{in the implicit form or} \\ &X(t) = X_A + S_x t, \qquad Y(t) = Y_A + S_y t \quad \text{in the parametric or} \\ &Y = kX + q, \quad \text{resp. } X = mY + p \quad \text{in the explicit form.} \end{aligned} \tag{7}$$

In the case of E^3 a line cannot be expressed in the implicit form, but as an intersection of two planes or in the parametric form as:

$$\begin{aligned} &a_1X + b_1Y + c_1 = 0 \quad \& \quad a_2X + b_2Y + c_2 = 0 \quad \text{in the implicit form or} \\ &X(t) = X_A + S_x t, \qquad Y(t) = Y_A + S_y t, \\ &Z(t) = Z_A + S_z t \quad \text{in the parametric form.} \end{aligned} \tag{8}$$

There is a special parametric form of the line in E^3, which uses the Plücker coordinates. It has a specific property as the point (X_A, Y_A, Z_A) is the closest point to the origin of the coordinate system (Blinn, 1977; Mahovsky and Wyvill, 2004; Platis and Theoharis, 2003; Wikipedia, 2020).

In computer graphics, some intersection algorithms are called clipping algorithms and serve to determine a part of one geometric entity inside the second one.

In the following, a brief classification of intersection algorithms in 2D and 3D will be presented with short characteristics; a short overview can be found in Wikipedia (2021a).

There are many variants of fundamental algorithms that differ in some aspects; mainly, the timing factor is the primary motivation. However, the claimed speed up mostly depends on the hardware properties (memory caching, processor used, etc.), programmer's skill and actual language and compiler used.

3. Intersection Algorithms in 2D

Algorithms for intersections of different 2D geometric entities have been studied for a long time from various aspects, primarily due to the computation speed, robustness and limited numerical precision of the floating-point representation. The majority of 2D algorithms deal with an intersection of a line or a half-line (ray) or a line segment with 2D geometric entity, e.g. a rectangle, convex polygon (Cyrus and Beck, 1978; Rappoport, 1991), non-convex polygon (Weiler and Atherton, 1977), quadric and cubic curves, parametric curves (Skala, 2021a) and areas with quadratic arcs (Skala, 2015, 1989, 1990a), etc.

There are two main strategies, which are "dual" in some sense:

- a position of the window, resp. polygon edges against the intersected line, resp. line segment, etc.,
- a position of the vertices of the window, resp. polygon against the intersected line, resp. line segment, etc.

3.1. *Intersection with a Rectangular Area*

Intersection algorithms with a rectangular area (window) are well known as the line clipping or as the line segment clipping algorithms. The first algorithm was developed and used for the flight simulator project led by Cohen (1969) in 1967. Efficient coding of the line segment position coding leading to significant computational reduction was introduced in Sproull and Sutherland (1968) and patented in 1972 (Sutherland, 1972). The Cohen-Sutherland algorithm is described in Newman and Sproull (1979), Comninos (2006), Matthes and Drakopoulos (2019a, 2019b), etc. The Cohen-Sutherland algorithm generates a bit-code LRTB, i.e. [Left, Right, Top, Bottom], for each end-point of the line segment, see Fig. 2. The coding is redundant. However, it enables simple identification of the cases, when the line segment is totally inside or outside as follows:

- **if** $(\mathbf{c}_A$ **lor** $\mathbf{c}_B) = [0000]$ **then** the line segment is totally inside,
- **if** $(\mathbf{c}_A$ **land** $\mathbf{c}_B) \neq [0000]$ **then** the line segment is totally outside,

where **land**, resp. **or** mean bit-wise *and*, resp. *or* operations.

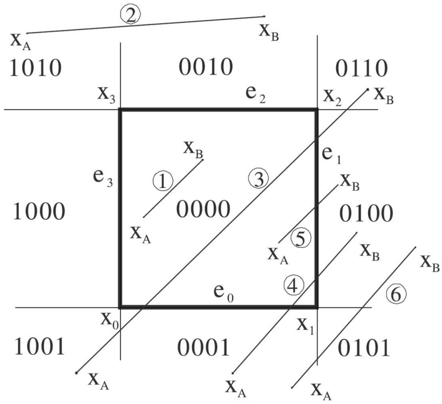

Fig. 2. Cohen-Sutherland coding.

Table 1
Numerical summation codes $C_{AB} = C_A + C_B$, IN – inside area, C – corner area, S – side area, n/a – non-applicable cases or outside case.

	C_{AB} / C_A	C_B	IN 0 0000	C 5 0101	S 4 0100	C 6 0110	S 2 0010	C 10 1010	S 8 1000	C 9 1001	S 1 0001
IN	0	0000	IN	5	4	6	2	10	8	9	1
C	5	0101	5	n/a	n/a	n/a	7	15	13	n/a	n/a
S	4	0100	4	n/a	n/a	n/a	6	14	12	13	5
C	6	0110	6	n/a	n/a	n/a	n/a	n/a	14	15	7
S	2	0010	2	7	6	n/a	n/a	n/a	10	11	3
C	10	1010	10	15	14	n/a	n/a	n/a	n/a	n/a	11
S	8	1000	8	13	12	14	10	n/a	n/a	n/a	9
C	9	1001	9	n/a	13	15	11	n/a	n/a	n/a	n/a
S	1	0001	1	n/a	5	7	3	11	9	n/a	n/a

The ultimately deep classification of all the possible cases using arithmetic operation with the codes was described in Skala (2021b), see Table 1 and Fig. 3. The C_{AB} value is the index to the **array of functions** representing each case.

Distinguishing all the cases leads to more efficient coding and efficient implementation (Skala, 2021b); specific cases are presented in Table 2.

The Cohen-Sutherland algorithm can also be extended to the 3D case, i.e. intersection of a line segment with a cube or right-angled parallelepiped.

The Cohen-Sutherland algorithm was improved by Nicholl *et al.* (1987). It uses the window corners position classification in relation to the line segment position, see Fig. 4. The Nicholl-Lee-Nicholl algorithm was improved by Bui and Skala (1998) using some additional classification of possible cases and extended to the E^3 case in Skala and Bui (2001).

The algorithms (Liang and Barsky, 1983) and (Dörr, 1990) are based on the direct intersection computation of a line with the polygon edges in the parametric form. Analy-

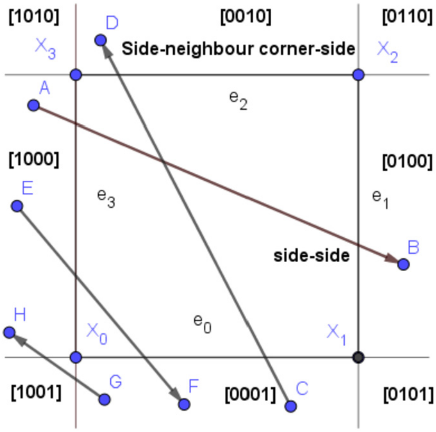

Fig. 3. Two specific situations – SS-SnCS: side-side and side-neighbour corner-side.

Table 2
Possible cases: n/a – non-applicable or solved by the C-S coding, C – corner area, S – side area, IN – inside area, End-points: IC – inside-corner, IS – inside-side; Cases: SS – side-side, SnCS – side-near corner-side, SdC – side-distant corner-side, CoC – corner-opposite corner, id – case re-indexing.

		id	−1	0	1	2	3	4	5	6	7
	Case		IN	C	S	C	S	C	S	C	S
		C_B	0	5	4	6	2	10	8	9	1
	C_A		0000	0101	0100	0110	0010	1010	1000	1001	0001
IN	0	0000	IN	IC	IS	IC	IS	IC	IS	IC	IS
C	5	0101	IC	n/a	n/a	n/a	SdC	CoC	SdC	n/a	n/a
S	4	0100	IS	n/a	n/a	n/a	SnCS	SdC	SS	SdC	SnCS
C	6	0110	IC	n/a	n/a	n/a	n/a	n/a	SdC	CoC	SdC
S	2	0010	IS	SdC	SnCS	n/a	n/a	n/a	SnCS	SdC	SS
C	10	1010	IC	CoC	SdC	n/a	n/a	n/a	n/a	n/a	SdC
S	8	1000	IS	SdC	SS	SdC	SnCS	n/a	n/a	n/a	SnCS
C	9	1001	IC	n/a	SdC	CoC	SdC	n/a	n/a	n/a	n/a
S	1	0001	IS	n/a	SnCS	SdC	SS	SdC	SnCS	n/a	n/a

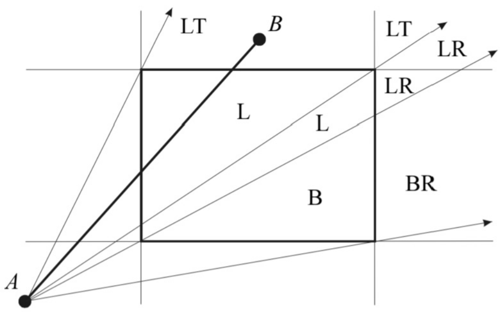

Fig. 4. Nicholl-Lee-Nicholl algorithm – window corners position evaluation.

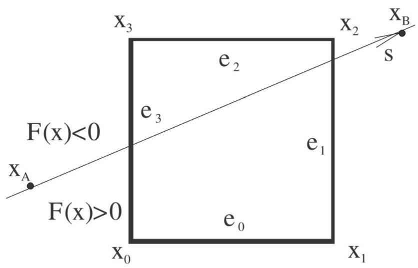

Fig. 5. Clipping against the rectangular window in E^2.

sis of the Nicholl-Lee-Nicholl and Liang-Barsky algorithms was given in Devai (2005). Simple and robust line and line segment clipping algorithms in E^2 was described in Skala (2004, 2005, 2012, 2020). They are based on the projective representation and homogeneous coordinates using a separation of the convex polygon vertices by the given line, see Fig. 5. The sign of the function values $F(\mathbf{x})$, which represents the given line, for each window corner gives a 4-bit code identifying the edges intersected by the given line. The algorithm can be extended for the convex polygon case.

3.2. *S-L-Clip Algorithm*

Let us consider an implicit function $F(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$, where $\mathbf{a} = [a, b : c]^T$ are coefficients of the given line p, $\mathbf{x} = [x, y : w]^T$ means a point on this line. Then the equation $F(\mathbf{x}) = 0$ represents the given line p in E^2 using the projective extension of the Euclidean space.

The clipping operation should determine the intersection points $\mathbf{x}_i = [x_i, y_i : w_i]^T$, $i = 1, 2$ of the given line with the window, if any. The line splits the plane into two parts, see Fig. 5. The corners of the window are split into two groups according to the sign of the function $F(\mathbf{x})$ value. This results into Smart-Line-Clip (S-L-Clip) algorithm, see Algorithm 1. It means that each corner can be classified by a bit value c_i as:

$$c_i = \begin{cases} 1, & F(\mathbf{x}_i) \geqslant 0, \\ 0, & \text{otherwise}, \end{cases} \qquad i = 0, \ldots, 3, \tag{9}$$

where $\mathbf{a} = [a, b : c]^T$ are coefficients of the given line p, $\mathbf{x} = [x, y : w]^T$ means a point on this line. Table 3 shows the codes for all situations (some of those are not possible). The **TAB1** and **TAB2** contain indices of edges of the window intersected by the given line (values in the **MASK** is used in the line segment algorithm).

It can be seen, that the S-L-Clip Algorithm 1 is quite simple and easily extensible for the convex polygon clipping case as well. Table 3 can be generated synthetically. It is significantly more straightforward than the algorithm (Liang and Barsky, 1984). It also supports SSE4 and GPU use directly and leads to simple implementations, as the cross-product and dot-product operations, are supported in hardware. It should be noted, that the algorithm is designed for a very general case, as the window corners and the points

Table 3
All cases; N/A – non-applicable (impossible) cases.

c	**c**	TAB1	TAB2	MASK	c	**c**	TAB1	TAB2	MASK
0	0000	None	None	None	15	1111	None	None	None
1	0001	0	3	0100	14	1110	3	0	None
2	0010	0	1	0100	13	1101	1	01	0100
3	0011	1	3	0010	12	1100	3	1	0010
4	0100	1	2	0010	11	1011	2	1	0010
5	0101	N/A	N/A	N/A	10	1010	N/A	N/A	N/A
6	0110	0	2	0100	9	1001	2	0	0100
7	0111	2	3	1000	8	1000	3	2	1000

Algorithm 1 S-L-Clip – line clipping algorithm by the rectangular window

procedure S-L-CLIP($\mathbf{x}_A, \mathbf{x}_B$); ▷ line is given by two points
  $\mathbf{p} := \mathbf{x}_A \wedge \mathbf{x}_B$; ▷ computation of the line coefficients
  for $i := 0$ **to** 3 **do**
    if $\mathbf{p}^T \mathbf{x}_i \geqslant 0$ **then** $c_i := 1$ **else** $c_i := 0$; ▷ codes computation
  end for
  if $\mathbf{c} \neq [0000]^T$ **and** $\mathbf{c} \neq [1111]^T$ **then** ▷ line intersects the window
    $i := TAB1[\mathbf{c}]$; $\mathbf{x}_A := \mathbf{p} \wedge \mathbf{e}_i$; ▷ first intersection point
    $j := TAB2[\mathbf{c}]$; $\mathbf{x}_B := \mathbf{p} \wedge \mathbf{e}_j$; ▷ second intersection point
    output($\mathbf{x}_A, \mathbf{x}_B$);
  end if
end procedure

defining the line, are generally in the projective representation, i.e. $w \neq 0$. Therefore, the S-L-Clip algorithm has further potential for optimization, especially for the case when the corner points of the window are given in the Euclidean coordinates, i.e. $w = 1$, and clipping is made in the Normalized Device Coordinate (NDC) system (Skala, 2020).

The modification of the S-L-Clip algorithm for a line segment clipping is simple and described in Skala (2004). The advantage of it is that the end-points and the window corners might be given generally in the projective space, i.e. $w \neq 0$. The cross-product is used for the intersection computation using SSE4 or GPU acceleration.

Other proposed modifications of algorithms can be found in Bui (1999), Andreev and Sofianska (1991), Bao and Peng (1996), Devai (2005, 2006, 1998), Duvalenko *et al.* (1990, 1993, 1996), Cai *et al.* (2001), Day (1992a, 1992b), Evangeline and Anitha (2014), Kaijian *et al.* (1990), Kodituwakku *et al.* (2013), Kong and Yin (2001), Maillot (1992), Wei *et al.* (2013), Slater and Barsky (1994), Ray (2012a, 2012b, 2014, 2015), Li (2016), Singh and Lumar (2016), Dev and Saharan (2019).

Some additional modifications of algorithms were published in Brackenbury (1984), Chao *et al.* (2009), Cheng and Yen (1989), Dimri (2015), Dimri *et al.* (2022), Elliriki *et al.* (2019), Hattab and Yusof (2014), Iraji *et al.* (2011), Jiang and Han (2013), Jianrong (2006), Kumar and Awasthi (2011), Kuzmin (1995), Li *et al.* (2014), Li and Lei (2012), Meriaux (1984), Molla *et al.* (2003), Nisha (2017b, 2017a), Sobkow *et al.* (1987), Sharma and Manohar (1993), Wang *et al.* (1998a, 1998b, 2012, 2001), Yang (1988), Pandey and Jain (2013), Bhuiyan (2009). The hardware FPGA implementation was proposed in Dawod (2011).

Analysis and comparisons of some clipping algorithms were published recently in Krammer (1992), Skala and Huy (2000), Skala *et al.* (1995), Nisha (2017a, 2017b), Matthes and Drakopoulos (2022), Ray (2012b).

3.3. *Intersection with Polygons*

Generic solutions for polygon clipping were developed by Weiler and Atherton (1977), Rappoport (1991), Vatti (1992), Wu *et al.* (2004), Xie *et al.* (2010), Zhang and Sabharwal (2002), Zhang *et al.* (2022). Boolean operations with polygons were introduced by Rivero and Feito (2000), Martinez *et al.* (2009).

Algorithms for a line clipping E^2 by a polygon depend on the polygon property, i.e. if the polygon is convex or non-convex. In the case of convex polygons, the convexity property and ordering of vertices enables to decrease complexity from $O(N)$ to $O(\lg N)$ (Skala, 1994). It should be noted that a similar complexity decrease is not possible in the E^3 case as there is no ordering.

In the non-convex polygon cases, when the polygon can be self-intersecting, etc., problems with robustness of computation can be expected. Also, in some cases a three-value logic is to be used in order to solve specific cases properly, e.g. a line passes a vertex, a line touches a vertex, a line lies on an edge, etc. (Mccoid and Gander, 2022; Skala, 1989, 1990a).

3.4. *Convex Polygons*

The Cyrus-Beck's algorithm (Cyrus and Beck, 1978) is probably the famous algorithm for line-convex polygon clipping. It is based on a computation of the parameter t of the given line in the parametric form with edges of the given convex polygon, Fig. 6. The algorithm is of $O(N)$ computational complexity and can be extended for the E^3 case.

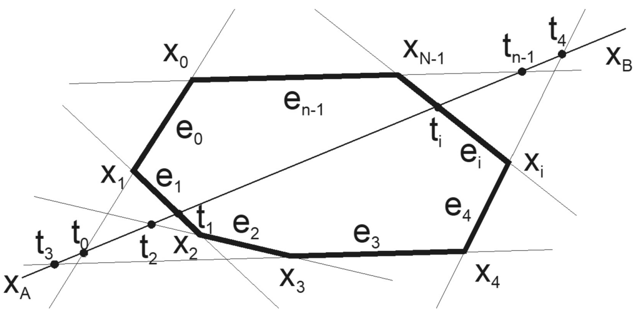

Fig. 6. Cyrus-Beck line clipping algorithm.

The Cyrus-Beck's algorithm is based on direct intersection computation of the given line p in the parametric form and a line on which the polygon edge e_i lies, see Fig. 6, in the implicit form, i.e. on a solution of two linear equations (vector notation is used):

$$\begin{aligned} p: \quad & \mathbf{x}(t) = \mathbf{x}_A + \mathbf{s}\,t, \\ e_i: \quad & \mathbf{n}_i^T\mathbf{x} + c_i = 0, \quad i = 0, \dots, N-1, \end{aligned} \tag{10}$$

where $\mathbf{x}_A = [x_A, y_A]^T$, $\mathbf{s} = [s_x, s_y]^T$ is the directional vector of the line p, $\mathbf{n}_i = [n_x, n_y]^T$ is the normal vector of the edge e_i.

Solving those equations, the parameter t for the intersection point is obtained as:

$$\mathbf{n}_i^T\mathbf{x}_A + \mathbf{n}_i^T\mathbf{s}\,t + c_i = 0. \tag{11}$$

Then t_i is the parameter t value for the intersection of the line p and the line on which the edge e_i lies, see Fig. 6.

$$t_i = -\frac{\mathbf{n}_i^T\mathbf{x}_A + c_i}{\mathbf{n}_i^T\mathbf{s}}. \tag{12}$$

It can be seen that the algorithm is not robust as if the line p is parallel or nearly parallel to the edge e_i, the expression $\mathbf{n}_i^T\mathbf{s} \to 0$ and $t_i \to \pm\infty$. The fraction computation might cause an overflow or high imprecision of the computed parameter t value, see Fig. 6.

It is hard to detect and solve such cases reliably and programmers usually use a sequence like

if $\left|\mathbf{n}_i^T\mathbf{s}\right| < eps$ **then** *a singular case*

which is an incorrect solution as the value *eps* is the programmer's choice and the value of $\mathbf{n}_i^T\mathbf{s}$ might also be close to the value of $\mathbf{n}_i^T\mathbf{x}_A + c_i$, see Eq. (12).

However, textbooks do not point out such dangerous construction as far as robustness and computational stability are concerned.

The modification of the Cyrus-Beck's algorithm using the cross product for more reliable detection of the "close to singular" cases was described by Skala (1993). Probably the most reliable modification of the Cyrus-Beck's algorithm is to use:

- a separation implicit function $F(\mathbf{x}) = 0$ representing the given line p defined as $F(\mathbf{x}) = \mathbf{n}^T\mathbf{x}_A + c$ for intersection detection as in Skala (2005),
- the parametric form of the given line for intersection computation with the found edges intersected, see Eq. (12).

The Cyrus-Beck's algorithm for a line clipping is described by Algorithm 2. It can be easily modified for the line segment clipping just restricting the range of the parameter t to $\langle 0, 1\rangle$, i.e.

$$\langle t_{\min}, t_{\max}\rangle := \langle t_{\min}, t_{\max}\rangle \cap \langle 0, 1\rangle. \tag{13}$$

Algorithm 2 Cyrus-Beck's line clipping algorithm

for $i := 0$ **to** $N-1$ **do**
  Compute $\mathbf{n}_i$ and c_i for all polygon edges
  ▷ pre-computation for the given convex polygon
end for
procedure C-B-CLIP($\mathbf{x}_A, \mathbf{x}_B$); ▷ line is given by two points
  $t_{\min} := -\infty$; $t_{\max} := \infty$; ▷ set initial conditions for the parameter t
  $\mathbf{s} := \mathbf{x}_B - \mathbf{x}_A$; ▷ computation of the line coefficients
  for $i := 0$ **to** $N-1$ **do** ▷ for each edge
    $q := \mathbf{n}_i^T \mathbf{s}$; ▷ pre-computation
    if $abs(q) < eps$ **then** NOP;
    ▷ Singular or close to singular case-usual solution
    else
      $t = -(\mathbf{n}_i^T \mathbf{x}_A + c_i)/\mathbf{n}_i^T \mathbf{s}$;
      if $q < 0$ **then** $t_{\min} := \max(t, t_{\min})$;
      else $t_{\max} := \min(t, t_{\max})$;
      end if
    end if
  end for ▷ all convex polygon edges processed
  if $t_{\min} < t_{\max}$ **then** ▷ intersection of a line and the polygon exists
    $\{\ \mathbf{x}_B := \mathbf{x}_A + \mathbf{s}\, t_{\max};\quad \mathbf{x}_A := \mathbf{x}_A + \mathbf{s}\, t_{\min};\ \}$
  end if
end procedure

It can be seen that the algorithm complexity is $O(N)$ and the division operation, which is the most consuming time operation in the floating-point representation, is used N times.[3]

However, only 2 values ($t_{\min}, t_{\max}$) of the parameter t are valid, i.e. $N-2$ computations of the parameter t are lost. Also, reliable detection of the "singular or close to singular" cases is difficult and time-consuming, especially in the E^3 case.

Some improvements and modifications were described by Skala (1993). As the edges of the convex polygon are ordered, the algorithm with the $O(\lg N)$ complexity was derived by Skala (1994). An algorithm based on space subdivision was described in Slater and Barsky (1994).

Another approach based on the implicit form of the given line and convex polygon vertices classification, the S-Clip algorithm, was developed in Skala (2021c) and modified by Konashkova (2014, 2015). Another algorithm based on the S-Clip algorithm was described in Skala (2021c). An algorithm for a line segment clipping based on the line segment end-points evaluation with the $O(N)$ complexity was described by Matthes and Drakopoulos (2022).

[3]There is a possibility to postpone division operations if the homogeneous coordinates are used, but comparison operations must be modified appropriately (Skala, 2020, 2021c).

The Liang-Barsky algorithm (Liang and Barsky, 1984, 1983) based on direct intersection computation of a line with the convex polygon edges in the parametric form has the $O(N)$ computational complexity, too.

The algorithm with the run-time $O(1)$ complexity using pre-computation was developed by Skala (1996b, 1996d). The algorithm was motivated by the scan-line raster conversion used recently for solving visibility in rendering. The memory requirements depend on the geometrical properties of the given convex polygon. A comparison of the $O(1)$ algorithm with the Cyrus-Beck algorithm is presented in Skala and Lederbuch (1996), Skala *et al.* (1996).

Other related algorithms or modifications of existing ones were published by: Li (2005), Nishita and Johan (1999), Raja (2019), Sun *et al.* (2006), Vatti (1992), Wang *et al.* (2005), Wijeweera *et al.* (2019), Sharma and Kaur (2016), Sharma and Manohar (1992) use the affine transformation.

3.5. *Non-Convex Polygons*

Probably, the first algorithm dealing with the non-convex polygon clipping was published in the Reentrant polygon clipping algorithm paper (Sutherland and Hodgman, 1974), followed by the Weiler-Atherton algorithm for polygon-polygon clipping (Weiler, 1980; Weiler and Atherton, 1977; Rappoport, 1991).

Intersections with arbitrary non-convex polygons were described in Greiner and Hormann (1998) and solutions of "the singular" (degenerated) cases were described in Foster *et al.* (2019). The algorithm (Skala, 1989) uses a three-value logic.

A robust solution of triangle-triangle intersection in E^2 is described in Mccoid and Gander (2022). Other algorithms or modifications are described in Dimri (2015), Evangeline and Anitha (2014), Lu and Wu (2002), Lu *et al.* (2002a), Tang and He (2009). The affine transformations are used in Huang (2013), Huang and Wangyong (2009), Huang and Liu (2002).

Algorithms that also handle arcs and use a three-value logic to handle singular cases properly, including self-intersecting non-convex polygons, were described in Skala (2015, 1989, 1990a), Wang and Chong (2016), Tran (1986).

Non-Polygonal Window

The algorithm for circular arc was described in Van Wyk (1984), Gupta *et al.* (2016), for overlapping areas by Li *et al.* (2012) and for circular window in Lu *et al.* (2002b), Kumar *et al.* (2018), Wu and Li (2022), Wu *et al.* (2006), Skala (1989), see Fig. 7. The above-mentioned algorithms lead to algorithms for set operations with polygons, i.e. union, intersection etc. of polygons described, e.g. Kui Liu *et al.* (2007), Martinez *et al.* (2009).

3.6. *Clipping Using Homogeneous Coordinates*

Homogeneous coordinates are used in computer graphics not only for geometric transformations. Sproull and Sutherland (1968) used the homogeneous coordinates in the Clipping divider in 1968. Arokiasamy (1989) used them with duality in 1989, Blinn (1991), Blinn

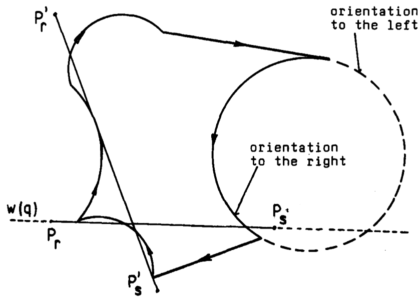

Fig. 7. Line clipping by an area with circular segments (taken from Skala, 1989).

and Newell (1978) described the clipping pipeline using the projective extension of the Euclidean space and Nielsen (1995) described the use of semi-homogeneous coordinates for clipping. New approach to 2D clipping based on the separation of the convex polygon vertices by the given line was presented in Kolingerová (1994, 1997) and Skala (2004, 2005, 2012, 2020).

In the following, algorithms related to the intersection in 3D will be briefly mentioned in a short introductory overview.

4. Intersection Algorithms in 3D

Intersection algorithms in 3D are widely used in many applications. An overview of the clipping algorithms is given in the Bui's PhD (Bui, 1999). The intersection of a line segment with a polygon in 3D was studied in Segura and Feito (1998) and the intersection of polygonal models was analysed by Melero *et al.* (2019). Algorithms for 3D clipping were overviewed in Skala (1990b) and reliable intersection tests with geometrical objects were published by Held (1998). Boolean operations with polygonal and polyhedral meshes were described by Landier (2017).

Line-Viewing Pyramid

Special attention was recently given to a line clipping by a pyramid in 3D due to the perspective pyramid clipping. The problem was analysed recently by Cohen (1969), Sproull and Sutherland (1968), Blinn (1991), Blinn and Newell (1978), Skala and Bui (2000, 2001).

Convex Polyhedron Case

The Cyrus-Beck's algorithm (Cyrus and Beck, 1978) is probably the famous algorithm for the line-convex polyhedron clipping in E^3. It computes a parameter t of a line in the parametric form and plane of the given face of the convex polyhedron. The algorithm is of the $O(N)$ computational complexity given by the fact that in the E^3 space there is "no order" of the polyhedron facets defined. Rogers and Rybak (1985) published a more general clipping algorithm in 3D in 1995.

The algorithm with the $O_{\exp}(\sqrt{N})$ complexity was described in Skala (1997, 2014). It assumes a triangular mesh, i.e. there is info on the neighbour triangles available. The algorithm is based on two planes representing the given line in E^3 and testing of the neighbours in the triangular mesh of the given polyhedron. The algorithm was modified by Konashkova (2015). An interesting approach using the vertex connection table was published in Konashkova (2014).

Using pre-computation, the algorithm in E^3 with a run-time $O(1)$ complexity was developed by Skala (1996c). Comparison was presented in Skala *et al.* (1996).

Ray-Convex Polyhedron

The Moeller-Trumbore algorithm for a ray-triangle intersection was published in Möller and Trumbore (1997). Since then many modifications and approaches have been published, e.g. Xiao *et al.* (2020) using GPUs, Skala (2010, 2008a) uses the computation of the barycentric coordinates in the homogeneous coordinates, Rajan *et al.* (2020) uses dual-precision fixed-point arithmetic for low-power ray-triangle intersections. Platis and Theoharis (2003) published an algorithm for a ray-tetrahedron intersection using the Plücker coordinates. The intersection with the AABBox is described in Eisemann *et al.* (2007), Kodituwakku and Wijeweera (2012), Maonica *et al.* (2017) and Mahovsky and Wyvill (2004). Other algorithms are available in Sharma and Manohar (1993), Skala (1996a), Williams *et al.* (2005), Llanas and Sainz (2012). The 3D line segment-triangle intersection algorithm is described in Jokanovic (2019), Amanatides and Choi (1995), Lagae and Dutré (2005) (in 2D only) and a ray/convex polyhedron intersection was described in Zheng and Millham (1991). Intersection of a line or a ray with a triangle using the SSE4 instructions was developed and described in Havel and Herout (2010). An extensive list of relevant publications can be found via Wikipedia (2021c).

Intersection with Complex Objects

The intersection computation with implicitly defined objects was published by Petrie and Mills (2020), intersection with a torus was published by Cychosz (1991) and alternative formulations were given in Skala (2013a). Reshetov (2022) published an efficient algorithm for a ray/ribbon intersections computation, ray tracing of 3D Bézier curves given by Reshetov (2017) and a ray/bilinear patch intersection (Reshetov, 2019). The intersection with general quadrics using the homogeneous coordinates was described in Skala (2015) and clipping by a spherical window was published by Deng *et al.* (2006).

However, as polygonal models are mostly formed by triangular surfaces, a special attention is also targeted to *triangle-triangle* intersections.

Triangle-Triangle Intersection in 3D

The computation of the intersection of triangles is probably the most important, as nearly all Computer Aided Design (CAD) systems depend on efficient, robust and reliable computation. Figures 8 and 9 present the non-trivial cases, when triangles are split into a set of triangles, which potentially leads to an explosion of small triangles and numerical and robustness problems.

In the CAD systems, two different data sets are usually used:

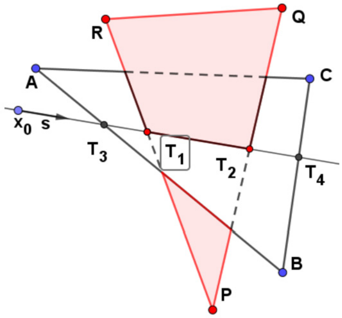

Fig. 8. Triangle-triangle intersection case-1.

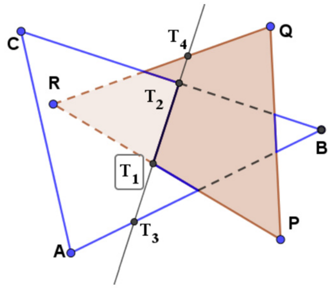

Fig. 9. Triangle-triangle intersection case-2.

- set of triangles – there is no connection between triangles; typical example is the STL format for the 3D print,
- triangular mesh – there is information on the neighbours of the given triangles and triangles sharing the given vertex; a typical example is the winged edge or the half-edge data structures, etc.

An efficient triangle-triangle intersection algorithm was developed by Möller (1997). It is based on the mutual triangle intersection with the plane of the other. Other methods or approaches were described by Chang and Kim (2009), Danaei *et al.* (2017), Devillers and Guigue (2002), Elsheikh and Elsheikh (2014), Guigue and Devillers (2003), Held (1998), Sabharwal and Leopold (2016), Sabharwal *et al.* (2013), Sabharwal and Leopold (2015), Shen *et al.* (2003), Tropp *et al.* (2006), Roy and Dasari (1998), Wei (2014), Ye *et al.* (2015). A deep analysis of possible situations is given in Lo and Wang (2004). Robust and reliable solution of the triangle-triangle intersection was developed by Mccoid and Gander (2022).

Clipping triangular strips using homogeneous coordinates was described by Maillot (1991) in GEM II (Arvo, 1991). Parallel exact algorithm for the intersection of large 3D triangular meshes was described in de Magalhães *et al.* (2020) and a comparison of triangle-triangle tests on GPU was described in Xiao *et al.* (2020). Triangular mesh repair was described by McLaurin *et al.* (2013).

5. Conclusion

This contribution briefly summarizes known clipping algorithms with some extent to the intersection in 3D and ray-tracing related algorithms. The list of published papers related to clipping algorithms should be complete to the author's knowledge and extensive search via Web of Science, Scopus, Research Gate and WEB search with the related topics. The relevant DOIs were included, if found. If other source was found, the relevant URL was included.

There is hope that this summary will help researchers, students and software developers to find relevant papers easily.

However, developers are urged to consider a limited precision of the floating-point representation (Wikipedia, 2021b) and handle numerical robustness issues properly for near *singular* cases in the actual implementations.

Surprisingly, during this summary preparation it was found that there are still some problems to be addressed and explored more deeply, like a robust and efficient intersection of triangular meshes as application of triangle-triangle intersection algorithms tend to lead to inconsistencies, inefficiency and unreliability, in general.

A short list of relevant books and research journals is given in Appendix A.

Appendix A

There are many books published related to intersection algorithms, clipping and computer graphics, which give more context and deeper understanding, e.g.:

- Salomon, D.: The Computer Graphics Manual – Salomon (2011),
- Salomon, D.: Computer Graphics and Geometric Modelling – Salomon (1999),
- Agoston, M.K.: Computer Graphics and Geometric Modelling: Mathematics – Agoston (2005),
- Agoston, M.K.: Computer Graphics and Geometric Modelling: Implementation & Algorithms – Agoston (2004),
- Lengyel, E.: Mathematics for 3D Game Programming and Computer Graphics – Lengyel (2011),
- Vince, J.: Introduction to the Mathematics for Computer Graphics – Vince (2010) (basic mathematical description of mathematics for undergraduates),
- Foley, J.D., van Dam, A., Feiner, S., Hughes, J.F.: Computer graphics – principles and practice – Foley *et al.* (1990),
- Hughes, J.F., van Dam, A., McGuire, M., Sklar, D.F., Foley, J.D., Feiner, S.K., Akeley, K.: Computer Graphics – Principles and Practice – Hughes *et al.* (2014),
- Ferguson, R.S.: Practical Algorithms for 3D Computer Graphics – Ferguson (2013),
- Shirley, P., Marschner, S.: Fundamentals of Computer Graphics – Shirley and Marschner (2009),
- Marschner, S., Shirley, P.: Fundamentals of Computer Graphics – Marschner and Shirley (2016),

- Theoharis, T., Platis, N., Papaioannou, G., Patrikalakis, N.: Graphics and Visualization: Principles & Algorithms – Theoharis *et al.* (2008),
- Comninos, P.: Mathematical and Computer Programming Techniques for Computer Graphics – Comninos (2005),
- Schneider, P.J., Eberly, D.H.: Geometric Tools for Computer Graphics – Schneider and Eberly (2003),
- Ammeraal, L., Zhang, K.: Computer graphics for Java programmers – Ammeraal and Zhang (2017),
- Vince, J.: Matrix Transforms for Computer Games and Animation – Vince (2012).

There are also computer graphics books using OpenGL interface, e.g.:

- Hill, F.S., Kelley, S.M.: Computer Graphics Using OpenGL – Hill and Kelley (2006),
- Angel, E., Shreiner, D.: Interactive Computer Graphics – Angel and Shreiner (2011),
- Hearn, D.D., Baker, M.P., Carithers, W.: Computer Graphics with OpenGL – Hearn *et al.* (2010),
- Govil-Pai, S.: Principles of Computer Graphics: Theory and Practice Using OpenGL and Maya – Govil-Pai (2005).

More advanced books using Geometric Algebra and Conformal Geometric Algebra approaches are recommended for a deeper study, e.g.:

- Vince, J.: Geometric Algebra: An Algebraic System for Computer Games and Animation – Vince (2009),
- Vince, J.: Geometric Algebra for Computer Graphics – Vince (2008),
- Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry – Dorst *et al.* (2009),
- Hildenbrand, D.: Foundations of Geometric Algebra Computing – Hildenbrand (2012),
- Kanatani, K.: Understanding Geometric Algebra: Hamilton, Grassmann, and Clifford for Computer Vision and Graphics – Kanatani (2015),
- Calvet, R.G.: Treatise of Plane Geometry through Geometric Algebra – Calvet (2007),
- Guo, H.: Modern Mathematics and Applications in Computer Graphics and Vision – Guo (2014),
- Lengyel, E: Mathematics for 3D Game Programming and Computer Graphics – Lengyel (2011),
- Salomon, D.: Transformations and Projections in Computer Graphics – Salomon (2006),
- Salomon, D.: The Computer Graphics Manual – Salomon (2006),
- Pharr, M., Jakob, W., Humphreys, G.: Physically Based Rendering: From Theory to Implementation – Pharr *et al.* (2016),
- Thomas, A.: Integrated Graphic and Computer Modelling – Thomas (2008) – describes the implementation of algorithms with examples with assembler codes.

It is also recommended to study "the historical" books, e.g.:

- Newman, W.M., Sproull, R.F.: Principles of Interactive Computer Graphics – Newman and Sproull (1979),

- Harrington, S.: Computer Graphics: A Programming Approach – Harrington (1987),
- Mortenson, M.E.: Computer Graphics: An Introduction to the Mathematics and Geometry – Mortenson (1988),
- Watt, A.: Fundamentals of Three-Dimensional Computer Graphics – Watt (1993),
- Akenine-Moller, T., Haines, E., Hoffman, N.: Real-Time Rendering – Akenine-Moller *et al.* (2008),
- Eberly, D.H.: Game Physics – Eberly (2003),
- Rogers, D.F., Adams, J.A.: Mathematical Elements for Computer Graphics – Rogers and Adams (1989).

Many algorithms with codes are presented in GEMS books:

- Graphics Gems, Ed. Glassner, A. – Glassner (1990),
- Graphics Gems II, Ed. Arvo, J. – Arvo (1991),
- Graphics Gems III, Ed. Kirk, D. – Kirk (1992),
- Graphics Gems IV, Ed. Heckbert, P.S. – Heckbert (1994).

and in the leading computer graphics journals:

- ACM Transactions on Graphics (TOG),
- Computer Graphics Forum (CGF),
- Computers & Graphics (C&G),
- IEEE Trans. on Visualization and Computer Graphics (TVCG),
- The Visual Computer (TVC),
- Computer Animation and Virtual Worlds (CAVW),
- Journal of Graphics Tools (JGT),
- Graphical Models.

The books mentioned above present a wide variety of intersection algorithm principles and applications.

Acknowledgements

The author would like to thank colleagues and students at the University of West Bohemia in Pilsen, VSB-Technical University and Ostrava University in Ostrava for their comments and recommendations, anonymous reviewers for hints and constructive suggestions.

The author is also grateful to several authors of recently published relevant papers for sharing their views and hints provided.

References

Agoston, M.K. (2004). *Computer Graphics and Geometric Modelling: Implementation & Algorithms*. Springer-Verlag, Berlin, Heidelberg. 1852338180.

Agoston, M.K. (2005). *Computer Graphics and Geometric Modelling: Mathematics*. Springer-Verlag, Berlin, Heidelberg. 1852338172.

Akenine-Moller, T., Haines, E., Hoffman, N. (2008). *Real-Time Rendering*, 3rd ed. A. K. Peters, Ltd., USA. 1568814240.

Amanatides, J., Choi, K.Y. (1995). Ray Tracing Triangular Meshes. http://www.cs.yorku.ca/~amana/research/mesh.pdf.

Ammeraal, L., Zhang, K. (2017). *Computer Graphics for Java Programmers*. Springer Cham. 978-3-319-63356-5 https://doi.org/10.1007/978-3-319-63357-2.

Andreev, R., Sofianska, E. (1991). New algorithm for two-dimensional line clipping. *Computers and Graphics*, 15(4), 519–526. https://doi.org/10.1016/0097-8493(91)90051-I.

Angel, E., Shreiner, D. (2011). *Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL*, 6th ed. Addison-Wesley Publishing Company, USA. 0132545233.

Arokiasamy, A. (1989). Homogeneous coordinates and the principle of duality in two dimensional clipping. *Computers and Graphics*, 13(1), 99–100. https://doi.org/10.1016/0097-8493(89)90045-9.

Arvo, J. (1991). *Graphics Gems II*. Academic Press Professional, Inc., USA. 0120644800.

Bao, H., Peng, Q. (1996). Efficient polygon clipping algorithm. *Zidonghua Xuebao/Acta Automatica Sinica*, 22(6), 741–744.

Bhuiyan, M.I. (2009). Designing a line-clipping algorithm by categorizing line dynamically and using intersection point method. In: *2009 International Conference on Electronic Computer Technology*, Macau, China, pp. 22–25. https://doi.org/10.1109/ICECT.2009.79.

Blinn, J.F. (1977). A homogeneous formulation for lines in 3 space. *ACM SIGGRAPH Computer Graphics*, 11(2), 237–241. https://doi.org/10.1145/965141.563900.

Blinn, J.F. (1991). A trip down the graphics pipeline: line clipping. *IEEE Computer Graphics and Applications*, 11(1), 98–105. https://doi.org/10.1109/38.67707.

Blinn, J.F., Newell, M.E. (1978). Clipping using homogeneous coordinates. In: *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1978*, pp. 245–251. https://doi.org/10.1145/800248.807398.

Brackenbury, I.F. (1984). Line clipping in interactive computer graphics. *IBM Technical Disclosure Bulletin*, 27(1B), 549–552.

Bui, D.H. (1999). *Algorithms for Line Clipping and Their Complexity* (supervisor: V. Skala). PhD thesis, University of West Bohemia, Pilsen. http://graphics.zcu.cz/files/DIS_1999_Bui_Duc_Huy.pdf.

Bui, D.H., Skala, V. (1998). Fast algorithms for clipping lines and line segments in E2. *Visual Computer*, 14(1), 31–37. https://doi.org/10.1007/s003710050121.

Cai, M., Yuan, C.-F., Song, J.-Q., Cai, S.-J. (2001). A fast line clipping algorithm for circular windows. *Journal of Computer-Aided Design and Computer Graphics*, 13(12), 1063–1067. https://doi.org/10.5121/ijcga.2018.8201.

Calvet, R.G. (2007). *Treatise of Plane Geometry Through Geometric Algebra*. Cerdanyola del Valles, Spain. 978-84-611-9149-9. http://www.xtec.cat/~rgonzal1/treatise-sample.pdf.

Chang, J.-W., Kim, M.-S. (2009). Technical section: efficient triangle-triangle intersection test for OBB-based collision detection. *Computers & Graphics*, 33(3), 235–240. https://doi.org/10.1016/j.cag.2009.03.009.

Chao, C., Zhaoyin, Z., Changsong, S. (2009). A midpoint segmentation clipping algorithm of circular window against line. In: *2009 International Forum on Computer Science-Technology and Applications*, Chongqing, China, pp. 15–19. https://doi.org/10.1109/IFCSTA.2009.10.

Cheng, F., Yen, Y.-k. (1989). A parallel line clipping algorithm and its implementation. In: Dew, P.M., Heywood, T.R., Earnshaw, R.A. (Eds.), *Parallel Processing for Computer Vision and Display*. Addison-Wesley, USA, pp. 338–350. 978-0201416053.

Cohen, D. (1969). *Incremental Methods for Computer Graphics*. Technical report, Harvard University, Cambridge, Massachusetts, USA. https://apps.dtic.mil/sti/pdfs/AD0694550.pdf.

Comninos, P. (2005). *Mathematical and Computer Programming Techniques for Computer Graphics*. Springer-Verlag, Berlin, Heidelberg. 1852339020. https://doi.org/10.1007/978-1-84628-292-8.

Comninos, P. (2006). Two-dimensional clipping. In: *Mathematical and Computer Programming Techniques for Computer Graphics*. Springer, London. https://doi.org/10.1007/978-1-84628-292-8_6.

Coxeter, H.S.M., Beck, G. (1992). *The Real Projective Plane*. Springer-Verlag, Berlin, Heidelberg. 0387978895.

Cychosz, J.M. (1991). Intersecting a ray with an elliptical torus. In: *Graphics Gems II*. Morgan Kaufmann, pp. 251–256. 9780080507545. https://doi.org/10.1016/B978-0-08-050754-5.50054-2.

Cyrus, M., Beck, J. (1978). Generalized two- and three-dimensional clipping. *Computers and Graphics*, 3(1), 23–28. https://doi.org/10.1016/0097-8493(78)90021-3.

Danaei, B., Karbasizadeh, N., Tale Masouleh, M. (2017). A general approach on collision-free workspace determination via triangle-to-triangle intersection test. *Robotics and Computer-Integrated Manufacturing*, 44, 230–241. https://doi.org/10.1016/j.rcim.2016.08.013.

Dawod, A.I. (2011). Hardware implementation of line clipping algorithm by using FPGA. *Tikrit Journal of Engineering Science*, 18, 89–105.

Day, J.D. (1992a). An algorithm for clipping lines in object and image space. *Computers and Graphics*, 16(4), 421–426. https://doi.org/10.1016/0097-8493(92)90029-U.

Day, J.D. (1992b). A new two dimensional line clipping algorithm for small windows. *Computer Graphics Forum*, 11(4), 241–245. https://doi.org/10.1111/1467-8659.1140241.

de Magalhães, S.V.G., Franklin, W.R., Andrade, M.V.A. (2020). An efficient and exact parallel algorithm for intersecting large 3-D triangular meshes using arithmetic filters. *Computer-Aided Design*, 120, 102801. https://doi.org/10.1016/j.cad.2019.102801. https://www.sciencedirect.com/science/article/pii/S0010448519305330.

Deng, W., Lu, G., Chen, L. (2006). New 3D line clipping algorithm against spherical surface window. In: *International Technology and Innovation Conference 2006 (ITIC 2006)*. IET, USA, pp. 894–898. 0-86341-696-9. https://doi.org/10.1049/cp:20060886.

Dev, D., Saharan, P. (2019). Implementation of efficient line clipping algorithm. *International Journal of Innovative Technology and Exploring Engineering*, 8(7), 295–298.

Devai, F. (1998). An analysis technique and an algorithm for line clipping. In: *Proceedings. 1998 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics*, London, UK, 1998, pp. 157–165. https://doi.org/10.1109/IV.1998.694214.

Devai, F. (2005). Analysis of the Nicholl-Lee-Nicholl algorithm. In: *Computational Science and Its Applications– ICCSA 2005*, *Lecture Notes in Computer Science*, Vol. 3480. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11424758_75.

Devai, F. (2006). A speculative approach to clipping line segments. In: *Computational Science and Its Applications – ICCSA 2006*, *Lecture Notes in Computer Science*, Vol. 3980. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11751540_15.

Devillers, O., Guigue, P. (2002). *Faster Triangle-Triangle Intersection Tests*. Technical Report RR-4488, INRIA. https://hal.inria.fr/inria-00072100.

Dimri, S.C. (2015). Article: a simple and efficient algorithm for line and polygon clipping in 2-D computer graphics. *International Journal of Computers and Applications*, 127(3), 31–34. https://doi.org/10.5120/ijca2015906352.

Dimri, S.C., Tiwari, U.K., Ram, M. (2022). An efficient algorithm to clip a 2D-polygon against a rectangular clip window. *Applied Mathematics-A Journal of Chinese Universities*, 37, 147–158. https://doi.org/10.1007/s11766-022-4556-0.

Dorst, L., Fontijne, D., Mann, S. (2009). *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 9780080553108. https://doi.org/10.1016/B978-0-12-374942-0.X0000-0.

Duvalenko, V.J., Robbins, W.E., Gyurcsik, R.S. (1990). Improving line segment clipping. *Dr Dobb's Journal*, 15(7), 36.

Duvanenko, V.J., Gyurcsik, R.S., Robbins, W.E. (1993). Simple and efficient 2D and 3D span clipping algorithms. *Computers and Graphics*, 17(1), 39–54. https://doi.org/10.1016/0097-8493(93)90050-J.

Duvanenko, V.J., Robbins, W.E., Gyurcsik, R.S. (1996). Line-segment clipping revisited. *Dr Dobb's Journal*, 21(1), 107.

Dörr, M. (1990). A new approach to parametric line clipping. *Computer Graphics (Pergamon)*, 14(3–4), 449–464. https://doi.org/10.1016/0097-8493(90)90067-8.

Eberly, D.H. (2003). *Game Physics*. Elsevier Science Inc., USA. 1558607404.

Eisemann, M., Magnor, M., Grosch, T., Müller, S. (2007). Fast ray/axis-aligned bounding box overlap tests using ray slopes. *Journal of Graphics Tools*, 12(4), 35–46. https://doi.org/10.1080/2151237X.2007.10129248.

Elliriki, M., Reddy, C., Anand, K. (2019). An efficient line clipping algorithm in 2D space. *International Arab Journal of Information Technology*, 16(5), 798–807. https://iajit.org/PDF/September%202019,%20No.%205/11103.pdf.

Elsheikh, A.H., Elsheikh, M. (2014). A reliable triangular mesh intersection algorithm and its application in geological modelling. *Engineering with Computers*, 30(1), 143–157. https://doi.org/10.1007/s00366-012-0297-3.

Evangeline, D., Anitha, S. (2014). 2D polygon clipping using shear transformation: an extension of shear based 2D line clipping. In: *2014 IEEE IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pp. 1379–1383. https://doi.org/10.1109/ICACCCT.2014.7019326.

Ferguson, R.S. (2013). *Practical Algorithms for 3D Computer Graphics*, 2nd ed. A. K. Peters, Ltd., USA. 1466582529.

Foley, J.D., van Dam, A., Feiner, S., Hughes, J.F. (1990). *Computer Graphics – Principles and Practice*, 2nd ed. Addison-Wesley, USA. 978-0-201-12110-0.

Foster, E.L., Hormann, K., Popa, R.T. (2019). Clipping simple polygons with degenerate intersections. *Computers & Graphics: X*, 2, 100007. https://doi.org/10.1016/j.cagx.2019.100007.

Glassner, A.S. (Ed.) (1990). *Graphics Gems*. Academic Press Professional, Inc., USA. http://inis.jinr.ru/sl/vol1/CMC/Graphics_Gems_1,ed_A.Glassner.pdf. 0122861695.

Govil-Pai, S. (2005). *Principles of Computer Graphics: Theory and Practice Using OpenGL and Maya*. Springer-Verlag, Berlin, Heidelberg. 0387955046.

Greiner, G., Hormann, K. (1998). Efficient clipping of arbitrary polygons. *ACM Transactions on Graphics*, 17(2), 71–83. https://doi.org/10.1145/274363.274364.

Guigue, P., Devillers, O. (2003). Fast and robust triangle-triangle overlap test using orientation predicates. *Journal of Graphics Tools*, 8(1), 25–32. https://doi.org/10.1080/10867651.2003.10487580.

Guo, H. (2014). *Modern Mathematics and Applications in Computer Graphics and Vision*. World Scientific Publ., Singapore. 978-9814449328. https://doi.org/0.1142/8703.

Gupta, R., Tripathi, V.K., Singh, K., Pathak, N.K., Rastogi, R. (2016). An innovative and easy approach for clipping curves along a circular window. In: *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, India, 2016, pp. 638–643. https://doi.org/10.1109/CICT.2016.132.

Harrington, S. (1987). *Computer Graphics: A Programming Approach*, 2nd ed. McGraw-Hill, Inc., USA. 0070267537.

Hattab, A.S.A., Yusof, Y. (2014). Line clipping based on parallelism approach and midpoint intersection. *AIP Conference Proceedings*, 1602, 371–374. 9780735412361. https://doi.org/10.1063/1.4882513.

Havel, J., Herout, A. (2010). Yet faster ray-triangle intersection (using SSE4). *IEEE Transactions on Visualization and Computer Graphics*, 16(3), 434–438. https://doi.org/10.1109/TVCG.2009.73.

Hearn, D.D., Baker, M.P., Carithers, W. (2010). *Computer Graphics with OpenGL*, 4th ed. Prentice Hall Press, USA. 0136053580.

Heckbert, P.S. (Ed.) (1994). *Graphics Gems IV*. Academic Press Professional, Inc., USA. 0123361559.

Held, M. (1998). ERIT: a collection of efficient and reliable intersection tests. *Journal of Graphics Tools*, 2(4), 25–44. https://doi.org/10.1080/10867651.1997.10487482.

Hildenbrand, D. (2012). *Foundations of Geometric Algebra Computing*. Springer Publishing Company, Inc., London. 3642317936. https://doi.org/10.1007/978-1-84628-997-2.

Hill, F.S., Kelley, S.M. (2006). *Computer Graphics Using OpenGL*, 3rd ed. Prentice-Hall, Inc., USA. 0131496700.

Huang, W. (2013). Line clipping algorithm of affine transformation for polygon. In: *Intelligent Computing Theories, ICIC 2013*, *Lecture Notes in Computer Science*, Vol. 7995. Springer, Berlin, Heidelberg, pp. 55–60. https://doi.org/10.1007/978-3-642-39479-9_7.

Huang, W., Wangyong (2009). A novel algorithm for line clipping. In: *Proceedings – 2009 International Conference on Computational Intelligence and Software Engineering, CiSE 2009*, pp. 1–5. https://doi.org/10.1109/CISE.2009.5366550.

Huang, Y.Q., Liu, Y.K. (2002). An algorithm for line clipping against a polygon based on shearing transformation. *Computer Graphics Forum*, 21(4), 683–688. https://doi.org/10.1111/1467-8659.00626.

Hughes, J.F., van Dam, A., McGuire, M., Sklar, D.F., Foley, J.D., Feiner, S.K., Akeley, K. (2014). *Computer Graphics – Principles and Practice*, 3rd ed. Addison-Wesley, USA. 978-0-321-39952-6.

Iraji, M.S., Mazandarani, A., Motameni, H. (2011). An efficient line clipping algorithm based on Cohen-Sutherland line clipping algorithm. *American Journal of Scientific Research*, 14(1), 65–71. https://www.researchgate.net/publication/275964580_An_Efficient_Line_Clipping_Algorithm_based_on_Cohen-Sutherland_Line_Clipping_Algorithm.

Jiang, B., Han, J. (2013). Improvement in the Cohen-Sutherland line segment clipping algorithm. In: *2013 IEEE International Conference on Granular Computing (GrC)*, Beijing, China, 2013, pp. 157–161. https://doi.org/10.1109/GrC.2013.6740399.

Jianrong, T. (2006). A new algorithm of polygon clipping against rectangular window based on the endpoint and intersection-point encoding. *Journal of Engineering Graphics*.

Johnson, M. (1996). Proof by duality: or the discovery of "New" theorems. *Mathematics Today*, December, 138–153.

Jokanovic, S. (2019). Two-dimensional line segment–triangle intersection test: revision and enhancement. *Visual Computer*, 35(10), 1347–1359. https://doi.org/10.1007/s00371-018-01614-1.

Kaijian, S., Edwards, J.A., Cooper, D.C. (1990). An efficient line clipping algorithm. *Computers and Graphics*, 14(2), 297–301. https://doi.org/10.1016/0097-8493(90)90041-U.

Kanatani, K. (2015). *Understanding Geometric Algebra: Hamilton, Grassmann, and Clifford for Computer Vision and Graphics*. A. K. Peters, Ltd., USA. 1482259508. https://doi.org/10.1201/b18273.

Kirk, D. (Ed.) (1992). *Graphics Gems III*. Academic Press Professional, Inc., USA. 0124096719.

Kodituwakku, R., Wijeweera, K.R. (2012). An efficient line clipping algorithm for 3D space. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(5).

Kodituwakku, S.R., Wijeweera, K.R., Chamikara, M.A.P. (2013). An efficient algorithm for line clipping in computer graphics programming. *Ceylon Journal of Science*, 17(1), 1–7. https://www.researchgate.net/publication/261288113_An_Efficient_Algorithm_for_Line_Clipping_in_Computer_Graphics_Programming.

Kolingerová, I. (1994). 3D-line clipping algorithms – a comparative study. *The Visual Computer*, 11(2), 96–104. https://doi.org/10.1007/BF01889980.

Kolingerová, I. (1997). Convex polyhedron-line intersection detection using dual representation. *Visual Computer*, 13(1), 42–49. https://doi.org/10.1007/s003710050088.

Konashkova, A.M. (2014). Line – convex polyhedron intersection using vertex connections table. *Applied Mathematical Sciences*, 8(21–24), 1177–1186. https://doi.org/10.12988/ams.2014.4133.

Konashkova, A.M. (2015). Modified Skala's plane tested algorithm for line-polyhedron intersection. *Applied Mathematical Sciences*, 9(61–64), 3097–3103. https://doi.org/10.12988/ams.2015.52169.

Kong, D.H., Yin, B.C. (2001). The improvement on the algorithm of Cohen-Surtherland line clipping. *CAD/GRAPHICS 2001*, 807–810. 7-5062-5137-X.

Krammer, G. (1992). A line clipping algorithm and its analysis. *Computer Graphics Forum*, 11(3), 253–266. https://doi.org/10.1111/1467-8659.1130253.

Kui Liu, Y., Qiang Wang, X., Zhe Bao, S., Gomboši, M., Žalik, B. (2007). An algorithm for polygon clipping, and for determining polygon intersections and unions. *Computers and Geosciences*, 33(5), 589–598. https://doi.org/10.1016/j.cageo.2006.08.008.

Kumar, J., Awasthi, A. (2011). Modified trivial rejection criteria in Cohen-Sutherland line clipping algorithm. In: *Advances in Computing, Communication and Control. ICAC3 2011*, *Communications in Computer and Information Science*, Vol. 125. Springer, Berlin, Heidelberg, pp. 1–10. https://doi.org/10.1007/978-3-642-18440-6_1.

Kumar, P., Patel, F., Kanna, R. (2018). An efficient line clipping algorithm for circular windows using vector calculus and parallelization. *International Journal of Computational Geometry and Applications*, 8(1/2), 01–08. https://doi.org/10.5121/IJCGA.2018.8201.

Kuzmin, Y.P. (1995). Bresenham's line generation algorithm with built-in clipping. *Computer Graphics Forum*, 14(5), 275–280. https://doi.org/10.1111/1467-8659.1450275.

Lagae, A., Dutré, P. (2005). An efficient ray-quadrilateral intersection test. *Journal of Graphics Tools*, 10(4), 23–32. https://doi.org/10.1080/2151237X.2005.10129208.

Landier, S. (2017). Boolean operations on arbitrary polygonal and polyhedral meshes. *CAD Computer Aided Design*, 85, 138–153. https://doi.org/10.1016/j.cad.2016.07.013.

Lengyel, E. (2011). *Mathematics for 3D Game Programming and Computer Graphics*, 3rd ed. Course Technology Press, Boston, MA, USA. 1435458869.

Li, H. (2016). Analysis and Implementation of Cohen_Sutherland Line Clipping Algorithm. In: *Proceedings of the 2016 International Conference on Sensor Network and Computer Engineering*. Atlantis Press, China, pp. 482–485. 978-94-6252-217-6. https://doi.org/10.2991/icsnce-16.2016.94.

Li, W. (2005). Bisearch-based line clipping algorithm against a convex polygonal window. *Journal of Computer-Aided Design and Computer Graphics*, 17(5), 962–965.

Li, Z., Lei, G. (2012). Modified Sutherland-Cohen line clipping algorithm (in Chinese). *Computer Engineering and Applications*, 48(34), 175. https://caod.oriprobe.com/articles/31582699/Modified_Sutherland_Cohen_line_clipping_algorithm.htm.

Li, Z.-Q., He, Y., Tian, Z.-J. (2012). Overlapping area computation between irregular polygons for its evolutionary layout based on convex decomposition. *Journal of Software*, 7(2), 485–492. https://doi.org/10.4304/jsw.7.2.485-492.

Li, Z., He, D., Wang, J., Wang, M. (2014). An improved algorithm of Cohen-Sutherland line clipping. *WIT Transactions on Information and Communication Technologies*, 49, 575–582. 9781845648558.

Liang, Y.-D., Barsky, B.A. (1983). An analysis and algorithm for polygon clipping. *Communications of the ACM*, 26(11), 868–877. https://doi.org/10.1145/182.358439.

Liang, Y.D., Barsky, B.A. (1984). A new concept and method for line clipping. *ACM Transactions on Graphics (TOG)*, 3(1), 1–22. https://doi.org/10.1145/357332.357333.

Llanas, B., Sainz, F.J. (2012). A local search algorithm for ray-convex polyhedron intersection. *Computational Optimization and Applications*, 51(2), 533–550. https://doi.org/10.1007/s10589-010-9354-2.

Lo, S., Wang, W. (2004). A fast robust algorithm for the intersection of triangulated surfaces. *Engineering with Computers*, 20, 11–21. https://doi.org/10.1007/s00366-004-0277-3.

Lu, G., Wu, X. (2002). Midpoint-subdivision line clipping algorithm based on filtering technique. *Journal of Computer-Aided Design and Computer Graphics*, 14(6), 513–517.

Lu, G., Wu, X., Peng, Q. (2002a). An efficient line clipping algorithm based on adaptive line rejection. *Computers and Graphics (Pergamon)*, 26(3), 409–415. https://doi.org/10.1016/S0097-8493(02)00084-5.

Lu, G., Xing, J., Tan, J. (2002b). New clipping algorithm of line against circular window with multi-encoding approach. *Journal of Computer-Aided Design and Computer Graphics*, 14(12), 1133–1137.

Mahovsky, J., Wyvill, B. (2004). Fast ray-axis aligned bounding box overlap tests with plucker coordinates. *Journal of Graphics Tools*, 9(1), 35–46. https://doi.org/10.1080/10867651.2004.10487597.

Maillot, P.-G. (1991). Three-dimensional homogeneous clipping of triangle strips. In: *Graphics Gems II*. Elsevier Inc., USA, pp. 219–231. 9780080507545. https://doi.org/10.1016/B978-0-08-050754-5.50050-5.

Maillot, P.-G. (1992). A new, fast method for 2D polygon clipping: analysis and software implementation. *ACM Transactions on Graphics (TOG)*, 11(3), 276–290. https://doi.org/10.1145/130881.130894.

Maonica, B., Das, P., Ramteke, P.B., Koolagudi, S.G. (2017). Selective cropper for geometrical objects in Open-Flipper. In: Satapathy, S.C., Bhateja, V., Joshi, A. (Eds.), *Proceedings of the International Conference on Data Engineering and Communication Technology*. Springer, Singapore, pp. 391–399. 978-981-10-1675-2. https://doi.org/10.1007/978-981-10-1675-2_39.

Marschner, S., Shirley, P. (2016). *Fundamentals of Computer Graphics*, 4th ed. A. K. Peters, Ltd., USA. 1482229390.

Martinez, F., Rueda, A.J., Feito, F.R. (2009). A new algorithm for computing Boolean operations on polygons. *Computers and Geosciences*, 35(6), 1177–1185. https://doi.org/10.1016/j.cageo.2008.08.009.

Matthes, D., Drakopoulos, V. (2019a). Another simple but faster method for 2D line clipping. *International Journal of Computer Graphics & Animation (IJCGA)*, 9(1–3). https://doi.org/10.5121/ijcga.2019.9301. https://aircconline.com/ijcga/V9N3/9319ijcga01.pdf.

Matthes, D., Drakopoulos, V. (2019b). A simple and fast line-clipping method as a scratch extension for computer graphics education. *Computer Science and Information Technology*, 7, 40–47. https://doi.org/10.13189/csit.2019.070202.

Matthes, D., Drakopoulos, V. (2022). Line clipping in 2D: overview, techniques and algorithms. *Journal of Imaging*, 8(10). https://doi.org/10.3390/jimaging8100286. https://www.mdpi.com/2313-433X/8/10/286.

Mccoid, C., Gander, M.J. (2022). A provably robust algorithm for triangle-triangle intersections in floating-point arithmetic. *ACM Transactions on Mathematical Software*, 48(2). https://doi.org/10.1145/3513264.

McLaurin, D., Marcum, D., Remotigue, M., Blades, E. (2013). Repairing unstructured triangular mesh intersections. *International Journal for Numerical Methods in Engineering*, 93(3), 266–275. https://doi.org/10.1002/nme.4385.

Melero, F.J., Aguilera, A., Feito, F.R. (2019). Fast collision detection between high resolution polygonal models. *Computers and Graphics (Pergamon)*, 83, 97–106. https://doi.org/10.1016/j.cag.2019.07.006.

Meriaux, M. (1984). A two-dimensional clipping divider. In: *Eurographics Conference Proceedings*. https://doi.org/10.2312/eg.19841031.

Molla, R., Jorquera, P., Vivo, R. (2003). Fixed-point arithmetic line clipping. In: *WSCG '2003 Proceedings*, pp. 93–96.

Mortenson, M.E. (1988). *Computer Graphics: An Introduction to the Mathematics and Geometry*. Industrial Press, Inc., USA. 0831111828.

Möller, T. (1997). A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2(2), 25–30. https://doi.org/10.1080/10867651.1997.10487472.

Möller, T., Trumbore, B. (1997). Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools*, 2(1), 21–28. https://doi.org/10.1080/10867651.1997.10487468.

Newman, W.M., Sproull, R.F. (1979). *Principles of Interactive Computer Graphics*, 2nd ed. McGraw-Hill, Inc., USA. 0070463387.

Nicholl, T.M., Lee, D.T., Nicholl, R.A. (1987). An efficient new algorithm for 2-D line clipping: its development and analysis. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987*, pp. 253–262. https://doi.org/10.1145/37401.37432.

Nielsen, H.P. (1995). Line clipping using semi-homogeneous coordinates. *Computer Graphics Forum*, 14(1), 3–16. https://doi.org/10.1111/1467-8659.1410003.

Nisha, A. (2017a). Comparison of various line clipping algorithms: review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 7(1). https://doi.org/10.23956/ijarcsse/V7I1/0149.

Nisha, A. (2017b). A review: comparison of line clipping algorithms in 3D space. *International Journal of Advanced Research (IJAR)*, 5(1). https://doi.org/10.21474/IJAR01/3022.

Nishita, T., Johan, H. (1999). A scan line algorithm for rendering curved tubular objects. In: *Proceedings. Seventh Pacific Conference on Computer Graphics and Applications*, Seoul, Korea (South), pp. 92–101. https://doi.org/10.1109/PCCGA.1999.803352.

Pandey, A., Jain, S. (2013). Comparison of various line clipping algorithm for improvement. *International Journal of Modern Engineering Research*, 3(1), 69–74.

Petrie, F., Mills, S. (2020). Real time ray tracing of analytic and implicit surfaces. In: *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, Wellington, New Zealand, pp. 1–6. https://doi.org/10.1109/IVCNZ51579.2020.9290653.

Pharr, M., Jakob, W., Humphreys, G. (2016). *Physically Based Rendering: From Theory to Implementation*, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 0128006455.

Platis, N., Theoharis, T. (2003). Fast ray-tetrahedron intersection using plucker coordinates. *Journal of Graphics Tools*, 8(4), 37–48. https://doi.org/10.1080/10867651.2003.10487593.

Raja, S.P. (2019). Line and polygon clipping techniques on natural images – a mathematical solution and performance evaluation. *International Journal of Image and Graphics*, 19(2). https://doi.org/10.1142/S0219467819500128.

Rajan, K., Hashemi, S., Karpuzcu, U., Doggett, M., Reda, S. (2020). Dual-precision fixed-point arithmetic for low-power ray-triangle intersections. *Computers and Graphics (Pergamon)*, 87, 72–79. https://doi.org/10.1016/j.cag.2020.01.006.

Rappoport, A. (1991). An efficient algorithm for line and polygon clipping. *The Visual Computer*, 7(1), 19–28. https://doi.org/10.1007/BF01994114.

Ray, B.K. (2012a). An alternative algorithm for line clipping. *Journal of Graphics Tools*, 16(1), 12–24. https://doi.org/10.1080/2151237X.2012.641824.

Ray, B.K. (2012b). A line segment clipping algorithm in 2D. *International Journal of Computer Graphics*, 3(2), 51–76.

Ray, B.K. (2014). A procedure to clip line segment. *International Journal of Computer Graphics*, 5(1), 9–19. https://doi.org/10.14257/ijcg.2014.5.1.02.

Ray, B.K. (2015). Line clipping against arbitrary polygonal window. *International Journal of Computer Graphics*, 6(1), 12–24. https://doi.org/10.14257/ijcg.2015.6.1.01.

Reshetov, A. (2017). Exploiting Budan-Fourier and Vincent's theorems for ray tracing 3D Bézier curves. In: *Proceedings of High Performance Graphics, HPG '17*. Association for Computing Machinery, New York, NY, USA. 9781450351010. https://doi.org/10.1145/3105762.3105783.

Reshetov, A. (2019). Cool patches: a geometric approach to ray/bilinear patch intersections. In: *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*, pp. 95–109. 978-1-4842-4427-2. https://doi.org/10.1007/978-1-4842-4427-2_8.

Reshetov, A. (2022). Ray/ribbon intersections. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5(3). https://doi.org/10.1145/3543862.

Rivero, M., Feito, F.R. (2000). Boolean operations on general planar polygons. *Computers and Graphics (Pergamon)*, 24(6), 881–896. https://doi.org/10.1016/S0097-8493(00)00090-X.

Rogers, D.F., Rybak, L.M. (1985). On an efficient general line-clipping algorithm. *IEEE Computer Graphics and Applications*, 5(1), 82–86. https://doi.org/10.1109/MCG.1985.276298.

Rogers, D.F., Adams, J.A. (1989). *Mathematical Elements for Computer Graphics*, 2nd ed. McGraw-Hill, Inc., USA. 0070535299.

Roy, U., Dasari, V.R. (1998). Implementation of a polygonal algorithm for surface–surface intersections. *Computers & Industrial Engineering*, 34(2), 399–412. https://doi.org/10.1016/S0360-8352(97)00276-3.

Sabharwal, C., Leopold, J., McGeehan, D. (2013). Triangle-triangle intersection determination and classification to support qualitative spatial reasoning. *Polibits*, 48, 13–22. https://doi.org/10.17562/PB-48-2.

Sabharwal, C.L., Leopold, J.L. (2015). A triangle-triangle intersection algorithm. *Computers and Graphics*, 5(11), 27–35. https://doi.org/10.5121/csit.2015.51003.

Sabharwal, C.L., Leopold, J.L. (2016). A generic design for implementing intersection between triangles in computer vision and spatial reasoning. In: Pal, R. (Ed.), *Innovative Research in Attention Modeling and Computer Vision Applications*. IGI Global, USA, p. 41. https://doi.org/10.4018/978-1-4666-8723-3.ch008.

Salomon, D. (1999). *Computer Graphics and Geometric Modeling*, 1st ed. Springer-Verlag, Berlin, Heidelberg. 0387986820.

Salomon, D. (2006). *Transformations and Projections in Computer Graphics*. Springer-Verlag, Berlin, Heidelberg. 1846283922.

Salomon, D. (2011). *The Computer Graphics Manual*. Springer, USA, pp. 1–1496. 978-0-85729-885-0. https://doi.org/10.1007/978-0-85729-886-7.

Schneider, P.J., Eberly, D.H. (2003). *Geometric Tools for Computer Graphics*, *The Morgan Kaufmann Series in Computer Graphics*. Morgan Kaufmann, San Francisco, pp. 1–1009. 978-1-55860-594-7. https://doi.org/10.1016/B978-1-55860-594-7.50025-4.

Segura, R.J., Feito, F.R. (1998). An algorithm for determining intersection segment-polygon in 3D. *Computers and Graphics (Pergamon)*, 22(5), 587–592. https://doi.org/10.1016/s0097-8493(98)00064-8.

Sharma, M., Kaur, J. (2016). An improved polygon clipping algorithm based on affine transformation. *Advances in Intelligent Systems and Computing*, 379(1), 783–792. https://doi.org/10.1007/978-81-322-2517-1_75.

Sharma, N.C., Manohar, S. (1992). Line clipping revisited: two efficient algorithms based on simple geometric observations. *Computers and Graphics*, 16(1), 51–54. https://doi.org/10.1016/0097-8493(92)90071-3.

Sharma, N.C., Manohar, S. (1993). Three dimensional line-clipping by systematic enumeration. *(IFIP Transactions B: Computer Applications in Technology*, 1(9), 225–232. 0444815643.

Shen, H., Heng, P.A., Tang, Z. (2003). A fast triangle-triangle overlap test using signed distances. *Journal of Graphics Tools*, 8(1), 17–23. https://doi.org/10.1080/10867651.2003.10487579.

Shirley, P., Marschner, S. (2009). *Fundamentals of Computer Graphics*, 3rd ed. A. K. Peters, Ltd., USA. 1568814690.

Singh, R., Lumar, A. (2016). RJ-ASHI algorithm: a new polygon/line clipping algorithm for 2D space. *International Journal of Advanced Research in Computer Science and Software Engineering*, 6, 215–219.

Skala, V. (1989). Algorithms for 2D line clipping. In: Hansmann, W., Hopgood, F.R.A., Straßer, W. (Eds.), *EG 1989 Proceedings*. Eurographics Association, The Netherlands. https://doi.org/10.2312/egtp.19891026.

Skala, V. (1990a). Algorithms for clipping quadratic arcs. In: Chua, T.S., Kunii, T.L. (Eds.), *CGI Proceedings*. Springer, Tokyo, pp. 255–268. https://doi.org/10.1007/978-4-431-68123-6_16.

Skala, V. (1990b). *Clipping Algorithm*. Habilitation thesis. University of West Bohemia, Pilsen (partially in Czech). http://afrodita.zcu.cz/~skala/EDU-PUB/Habilitace-komplet.pdf.

Skala, V. (1993). An efficient algorithm for line clipping by convex polygon. *Computers and Graphics*, 17(4), 417–421. https://doi.org/10.1016/0097-8493(93)90030-D.

Skala, V. (1994). $O(lg\ N)$ line clipping algorithm in E^2. *Computers and Graphics*, 18(4), 517–524. https://doi.org/10.1016/0097-8493(94)90064-7.

Skala, V. (1996a). An efficient algorithm for line clipping by convex and non-convex polyhedra in E^3. *Computer Graphics Forum*, 15(1), 61–68. https://doi.org/10.1111/1467-8659.1510061.

Skala, V. (1996b). Line clipping in E^2 with $O(1)$ processing complexity. *Computer Graphics (Pergamon)*, 20(4), 523–530. https://doi.org/10.1016/0097-8493(96)00024-6.

Skala, V. (1996c). Line clipping in E^3 with expected complexity $O(1)$. *Machine Graphics and Vision*, 5(4), 551–562. https://doi.org/10.48550/arXiv.2201.00592.

Skala, V. (1996d). Trading time for space: an $O(1)$ average time algorithm for point-in-polygon location problem: theoretical fiction or practical usage? *Machine Graphics and Vision*, 5(3), 483–494.

Skala, V. (1997). A fast algorithm for line clipping by convex polyhedron in E^3. *Computers and Graphics (Pergamon)*, 21(2), 209–214. https://doi.org/10.1016/s0097-8493(96)00084-2.

Skala, V. (2004). A new line clipping algorithm with hardware acceleration. In: *Proceedings of Computer Graphics International Conference, CGI*, pp. 270–273. https://doi.org/10.1109/CGI.2004.1309220.

Skala, V. (2005). A new approach to line and line segment clipping in homogeneous coordinates. *Visual Computer*, 21(11), 905–914. https://doi.org/10.1007/s00371-005-0305-3.

Skala, V. (2008a). Barycentric coordinates computation in homogeneous coordinates. *Computers and Graphics (Pergamon)*, 32(1), 120–127. https://doi.org/10.1016/j.cag.2007.09.007.

Skala, V. (2008b). Intersection computation in projective space using homogeneous coordinates. *International Journal of Image and Graphics*, 8(4), 615–628. https://doi.org/10.1142/S021946780800326X.

Skala, V. (2010). Duality, barycentric coordinates and intersection computation in projective space with GPU support. *WSEAS Transactions on Mathematics*, 9(6), 407–416. http://afrodita.zcu.cz/~skala/PUBL/PUBL_2010/2010_NAUN-journal.pdf.

Skala, V. (2012). S-clip E^2: a new concept of clipping algorithms. In: *SIGGRAPH Asia Posters, SA '12*, pp. 1–2. https://doi.org/10.1145/2407156.2407200.

Skala, V. (2013a). Line-torus intersection for ray tracing: alternative formulations. *WSEAS Transactions on Computers*, 12(7), 288–297. https://doi.org/10.48550/ARXIV.2301.03191.

Skala, V. (2013b). Summation problem revisited – more robust computation. In: *17th International Conference on Computers – Recent Advances in Computer Science CSCC '13*, pp. 56–64. 978-960-474-311-7. https://doi.org/10.48550/arXiv.2211.04402.

Skala, V. (2014). Algorithms for line and plane intersection with a convex polyhedron with $O(sqrt(N))$ expected complexity in E3. In: *SIGGRAPH Asia 2014 Posters, SA '14*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2668975.2668976.

Skala, V. (2015). A new approach to line – sphere and line – quadrics intersection detection and computation. *AIP Conference Proceedings*, 1648, 1–4. 9780735412873. https://doi.org/10.1063/1.4913058.

Skala, V. (2020). Optimized line and line segment clipping in E2 and geometric algebra. *Annales Mathematicae et Informaticae*, 52, 199–215. https://doi.org/10.33039/ami.2020.05.001.

Skala, V. (2021a). Efficient intersection computation of the Bezier and Hermite curves with axis aligned bounding box. *WSEAS Transactions on Systems*, 20, 320–323. https://doi.org/10.37394/23202.2021.20.36.

Skala, V. (2021b). A new coding scheme for line segment clipping in E^2. In: *Computational Science and Its Applications – ICCSA 2021*, *Lecture Notes in Computer Science*, Vol. 12953, pp. 16–29. https://doi.org/10.1007/978-3-030-86976-2_2.

Skala, V. (2021c). A novel line convex polygon clipping algorithm in E^2 with parallel processing modification. In: *Computational Science and Its Applications – ICCSA 2021*, *Lecture Notes in Computer Science*, Vol. 12953. pp. 3–15. https://doi.org/10.1007/978-3-030-86976-2_1.

Skala, V. (2022). *Clipping and Intersection Algorithms: Short Survey and References*. arXiv: https://doi.org/10.48550/ARXIV.2206.13216. https://arxiv.org/abs/2206.13216.

Skala, V., Bui, D.H. (2000). Faster algorithm for line clipping against a pyramid in E3. *Machine Graphics and Vision*, 9(4), 841–850. https://doi.org/10.48550/arXiv.2201.00587.

Skala, V., Bui, D.H. (2001). Extension of the Nicholls-Lee-Nichols algorithm to three dimensions. *Visual Computer*, 17(4), 236–242. https://doi.org/10.1007/s003710000094.

Skala, V., Huy, B.D. (2000). Two new algorithms for line clipping in E2 and their comparison. *Machine Graphics and Vision*, 9(1/2), 297–306. https://doi.org/10.48550/arXiv.2201.00590.

Skala, V., Lederbuch, P. (1996). A comparison of a new O(1) and the cyrus-beck line clipping algorithms in E2. In: *Compugraphics'96: Fifth International Conference on Computational Graphics and Visualization Techniques*. ACM, Portugal, pp. 281–287. 972-8342-01-2. https://dspace5.zcu.cz/handle/11025/11808.

Skala, V., Kolingerova, I., Blaha, P. (1995). A comparison of 2D line clipping algorithms. *Machine Graphics and Vision*, 3(4), 625–633.

Skala, V., Lederbuch, P., Sup, B. (1996). A comparison of O(1) and Cyrus-Beck line clipping algorithm in E2 and E3. In: *SCCG96 Conference Proceedings*. Comenius University, Slovakia, pp. 17–44. https://doi.org/10.48550/arXiv.2111.07987. https://dspace5.zcu.cz/handle/11025/11806.

Slater, M., Barsky, B.A. (1994). 2D line and polygon clipping based on space subdivision. *The Visual Computer*, 10(7), 407–422. https://doi.org/10.1007/BF01900665.

Sobkow, M.S., Pospisil, P., Yang, Y.-H. (1987). A fast two-dimensional line clipping algorithm via line encoding. *Computers and Graphics*, 11(4), 459–467. https://doi.org/10.1016/0097-8493(87)90061-6.

Sproull, R.F., Sutherland, I.E. (1968). A clipping divider. In: *Fall Joint Computer Conference Proceedings, of the December 9–11, 1968, AFIPS '68 (Fall, part I)*. Association for Computing Machinery, New York, NY, USA, pp. 765–775. 9781450378994. https://doi.org/10.1145/1476589.1476687.

Stolfi, J. (1991). *Oriented Projective Geometry*. Academic Press Professional, Inc., USA. 0126720258.

Sun, C., Wang, W., Li, J., Wu, E. (2006). Line clipping against a polygon through convex segments. *Journal of Computer-Aided Design and Computer Graphics*, 18(12), 1799–1805.

Sutherland, I.E. (1972). Display windowing by clipping. *Google Patents*. https://patents.google.com/patent/US3639736A/en.

Sutherland, I.E., Hodgman, G.W. (1974). Reentrant polygon clipping. *Communications of the ACM*, 17(1), 32–42. https://doi.org/10.1145/360767.360802.

Tang, L.-L., He, Y.-J. (2009). A linear time algorithm for the line clipping against concave polygon. In: *Proceedings – 2009 International Conference on Information Engineering and Computer Science, ICIECS 2009*, pp. 1–4. https://doi.org/10.1109/ICIECS.2009.5364626.

Theoharis, T., Platis, N., Papaioannou, G., Patrikalakis, N.M. (2008). *Graphics and Visualization: Principles & Algorithms*, 1st ed. A. K. Peters/CRC Press, New York. https://doi.org/10.1201/b10676.

Thomas, A. (2008). *Integrated Graphics and Computer Modelling*, 1st ed. Springer, London. 1848001789.

Tran, C.-H. (1986). *Fast Clipping Algorithms for Computer Graphics*. PhD thesis, University of British Columbia. https://doi.org/10.14288/1.0096928. https://open.library.ubc.ca/collections/ubctheses/831/items/1.0096928.

Tropp, O., Tal, A., Shimshoni, I. (2006). A fast triangle to triangle intersection test for collision detection. *Computer Animation and Virtual Worlds*, 17(5), 527–535. https://doi.org/10.1002/cav.115.

Van Wyk, C.J. (1984). Clipping to the boundary of a circular-arc polygon. *Computer Vision, Graphics, and Image Processing*, 25(3), 383–392. https://doi.org/10.1016/0734-189X(84)90202-0.

Vatti, B.R. (1992). A generic solution to polygon clipping. *Communications of the ACM*, 35(7), 56–63. https://doi.org/10.1145/129902.129906.

Vince, J. (2009). *Geometric Algebra: An Algebraic System for Computer Games and Animation*, 1st ed. Springer, London. 1848823789.

Vince, J. (2010). *Introduction to the Mathematics for Computer Graphics*, 3rd ed. Springer-Verlag, Berlin, Heidelberg. 1849960224. https://link.springer.com/book/10.1007/978-1-4471-6290-2#toc.

Vince, J. (2012). *Matrix Transforms for Computer Games and Animation*. Springer, London. 1447143205.

Vince, J.A. (2008). *Geometric Algebra for Computer Graphics*, 1st ed. Springer-Verlag TELOS, Santa Clara, CA, USA. 1846289963. https://doi.org/10.1007/978-1-84628-997-2.

Wang, H., Chong, S. (2016). A high efficient polygon clipping algorithm for dealing with intersection degradation. *Dongnan Daxue Xuebao (Ziran Kexue Ban)/Journal of Southeast University (Natural Science Edition)*, 46(4), 702–707. https://doi.org/10.3969/j.issn.1001-0505.2016.04.005.

Wang, H., Wu, R., Cai, S. (1998a). A new algorithm for two-dimensional line clipping via geometric transformation. *Journal of Computer Science and Technology*, 13(5), 410–416. https://doi.org/10.1007/bf02948499.

Wang, H., Wu, R., Cai, S. (1998b). New efficient line clipping algorithm based on geometric transformation. *Ruan Jian Xue Bao/Journal of Software*, 9(10), 728–733.

Wang, J., Lu, G.-D., Peng, Q.-S., Wu, X.-H. (2005). Line clipping against polygonal window algorithm based on the multiple virtual boxes rejecting. *Journal of Zhejiang University: Science*, 6(Suppl 1), 100–107. https://doi.org/10.1631/jzus.2005.AS0100.

Wang, J., Cui, C., Gao, J. (2012). An efficient algorithm for clipping operation based on trapezoidal meshes and sweep-line technique. *Advances in Engineering Software*, 47(1), 72–79. https://doi.org/10.1016/j.advengsoft.2011.12.003.

Wang, X., Xue, Y., Fang, F., Chen, G. (2001). From probability model to a fast line clipping algorithm. In: *CAD/GRAPHICS 2001*, pp. 802–806.

Watt, A. (1993). *3d Computer Graphics*, 2nd ed. Addison-Wesley Longman Publishing Co., Inc., USA. 0201631865.

Wei, L.-Y. (2014). A faster triangle-to-triangle intersection test algorithm. *Computer Animation and Virtual Worlds*, 25(5–6), 553–559. https://doi.org/10.1002/cav.1558.

Wei, W., Ma, P., Lin, W. (2013). An improved Cohen-Sutherland region encoding algorithm. *Applied Mechanics and Materials*, 239–240, 1313–1317. 9783037855454. https://doi.org/10.4028/www.scientific.net/AMM.239-240.1313.

Weiler, K. (1980). Polygon comparison using a graph representation. In: *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '80*, pp. 10–18. https://doi.org/10.1145/800250.807462.

Weiler, K., Atherton, P. (1977). Hidden surface removal using polygon area sorting. In: *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '77*, pp. 214–222. https://doi.org/10.1145/563858.563896.

Wijeweera, K.R., Kodituwakku, S.R., Pathum Chamikara, M.A. (2019). A novel and efficient approach for line segment clipping against a convex polygon. *Ruhuna Journal of Science*, 10(2), 161–173. https://doi.org/10.4038/rjs.v10i2.81.

Wikipedia (2020). Plücker Matrix – Wikipedia, The Free Encyclopedia. [Online; accessed 12-May-2022]. https://en.wikipedia.org/wiki/Plucker_matrix.

Wikipedia (2021a). Clipping (Computer Graphics) – Wikipedia, The Free Encyclopedia. [Online; accessed 28-July-2021]. https://en.wikipedia.org/wiki/Clipping_(computer_graphics).

Wikipedia (2021b). IEEE 754 – Wikipedia, The Free Encyclopedia. [Online; accessed 11-July-2021]. https://en.wikipedia.org/wiki/IEEE_754.

Wikipedia (2021c). Ray Tracing (Graphics) – Wikipedia, The Free Encyclopedia. [Online; accessed 3-August-2021]. https://en.wikipedia.org/wiki/Ray_tracing_(graphics).

Williams, A., Barrus, S., Morley, R.K., Shirley, P. (2005). An efficient and robust ray-box intersection algorithm. In: *ACM SIGGRAPH 2005 Courses*. ACM, New York, NY, USA, p. 9. https://doi.org/10.1145/1198555.1198748.

Wu, Q., Huang, X., Han, Y. (2006). A clipping algorithm for parabola segments against circular windows. *Computers & Graphics*, 30(4), 540–560. https://doi.org/10.1016/j.cag.2006.03.001. https://www.sciencedirect.com/science/article/pii/S0097849306000732.

Wu, Y., Li, X. (2022). Curve intersection based on cubic hybrid clipping. *Visual Computing for Industry, Biomedicine, and Art*, 5(1). https://doi.org/10.1186/s42492-022-00114-3.

Wu, Z., Gou, C., Yang, D., Luo, Z. (2004). Line clipping algorithm against arbitrary polygons. *Journal of Computer-Aided Design and Computer Graphics*, 16(2), 228–233.

Xiao, L., Mei, G., Cuomo, S., Xu, N. (2020). Comparative investigation of GPU-accelerated triangle-triangle intersection algorithms for collision detection. *Multimedia Tools and Applications*, 81, 3165–3180. https://doi.org/10.1007/s11042-020-09066-3.

Xie, L., Li, P., Zhou, M., Wang, X. (2010). An clipping general polygons in regular girds algorithm base on successive encoding. In: *2010 International Conference on Computer Application and System Modeling, ICCASM 2010*, Taiyuan, 2010, pp. 4709–4713. https://doi.org/10.1109/ICCASM.2010.5619427.

Yang, W. (1988). New approach to line clipping in computer graphics display. *Zhongnan Kuangye Xueyuan Xuebao*, 18(1), 73–78.

Ye, X., Huang, L., Wang, L., Xing, H. (2015). An improved algorithm for triangle to triangle intersection test. In: *ICIA 2015 Proceedings*, pp. 2689–2694. https://doi.org/10.1109/ICInfA.2015.7279740.

Zhang, M., Sabharwal, C.L. (2002). An efficient implementation of parametric line and polygon clipping algorithm. In: *Proceedings of the ACM Symposium on Applied Computing*, pp. 796–800. https://doi.org/10.1145/508791.508945.

Zhang, Z., Fan, J., Xu, S., Chen, Z. (2022). VCS optimization method of Vatti algorithm for polygon overlay and parallelization using GPU. *Journal of Geo-Information Science*, 24(3), 437–447 (in Chinese). https://doi.org/10.12082/dqxxkx.2022.210409.

Zheng, J.L., Millham, C.B. (1991). A linear programming method for ray-convex polyhedron intersection. *Computers and Graphics*, 15(2), 195–204. https://doi.org/10.1016/0097-8493(91)90073-Q.

V. Skala is a full professor of computer science at the University of West Bohemia, Pilsen, Czech Republic. He received his Ing. (equivalent of MSc) degree in 1975 from the Institute of Technology in Pilsen, CSc. (equivalent of PhD) degree from the Czech Technical University in Prague, in 1981. In 1996, he became a full professor in computer science. He is a Fellow of the Eurographics Association, member of several editorial boards of international research journals and the editor-in-chief of the *Journal of WSCG* and *Computer Science Research Notes*. He is the organizer of the WSCG conferences on computer graphics, visualization and computer vision (www.wscg.eu) held annually since 1992.

His current research interests are computer graphics and visualization, applied mathematics, especially geometrical algebra, algorithms, and data structures.