WSEAS Trans.on Computers, Vol.11, No.9, ISSN 1109-2742, E-ISSN 2224-2872, pp.275-291, 2012

Geometry, Duality and Robust Computation in Engineering

VACLAV SKALA
Department of Computer Science and Engineering
Faculty of Applied Sciences, University of West Bohemia
Univerzitni 8, CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz http://www.VaclavSkala.eu

Abstract: - Robustness of computations in engineering is one of key issues as it is necessary to solve technical problems leading to ill conditioned solutions. Therefore the robustness and numerical stability is becoming a key issue more important that the computational time. In this paper we will show selected computational issues in numerical precision, well known cases of failures in computations. The Euclidean representation is used in today's computations, however the projective space (an extension of the Euclidean space) representation leads to more compact and robust formulations and to matrix-vector operations supported in hardware, e.g. by GPU.

Key-Words: - Euclidean space, projective space, homogeneous coordinates, duality, intersections, barycentric coordinates, planes intersection, Plucker coordinates, numerical precision.

1 Introduction

Data processing is one of the main fields in computer science. Data processing itself can be split to two main areas:

- processing of textual data
- processing of numerical data

Nowadays, computers use binary system for information and data representation. We use octal or hexadecimal representation for data representation. If we would be direct descendants of tetrapods we would have a great advantage as they had 8 fingers on a hand, see Fig.1. However, we have 5 fingers at a hand and use a decimal numeral system and for computation we use numbers with a decimal point, rational and irrational ones.

The first tetrapods had eight fingers on each hand

Figure 1

The above mentioned main two areas are quite different, but have many common algorithms, e.g. hashing. In the case of textual data we have "unlimited" dimensionality ("unlimited" length of a string) but limited interval of values (usually given by a number of symbols in the given alphabet). On the contrary in the case of numerical or geometrical data we have a limited dimensionality (usually 2 or 3 in the case of E^2 or E^3) but "unlimited" interval of values (usually $(-\infty, \infty)$). In the case of hashing techniques it lead us to a "unified" approach of hashing, but different construction and specification of the hash function used [8], [36], [37].

	Name	Base	Digits	E min	E max
BINARY					
B 16	Half	2	10+1	−14	15
B 32	Single	2	23+1	−126	127
B 64	Double	2	52+1	−1022	1023
B 128	Quad	2	112+1	−16382	16383
DECIMAL					
D 32		10	7	−95	96
D 64		10	16	−383	384
D 128		10	34	−6143	6144

IEEE 758-2008 standard

Table 1

Numerical data processing and numerical computations bring quite significant difficulties due to the limited precision of a real number representation as the floating point representation offers only a limited length of a mantissa and exponent. The standard IEEE754 specification offers the following formats, see Tab.1. It should be noted however that not all the modes are supported in many cases on different CPUs. Today's programming standard languages do not offer constructions for computation with an "arbitrarily" long integers or "unlimited" mantissa length (Algol 68 had a construction **long** that could extend the basic data type, e.g. **long** ..... **long real** etc.). Unfortunately it leads to numerical problems and possibly to disasters in engineering applications. There are also problems connected to uncontrolled overflow, infinities and NaN results [49].

There are several attempts, like logarithmic number representation [16] or continued fraction computation [2], [14]. However those approaches do not solve the principal problem as well.

It should be noted that the majority of computer science students are NOT AWARE of those aspects at all.

2 Numerical Precision and Robustness

Numerical data processing and numerical computation is the heart of nearly all engineering problems solution. On the other hand it seems to that in the engineering courses there is no attention given to the numerical precision in connection with the robustness of algorithms.

From the floating point representation it can be seen that the absolute precision depends on the actual exponent significantly, as the precision is given by the length of the mantissa **multiplied by the exponent**. As the mantissa is of the given length, not all numbers even rational numbers can be represented in a computer; of course irrational numbers cannot be stored in any case. It means that a value *x* is somehow modified in order to fit into the actual floating point representation. It means that a stored value *x* represents actually an interval [*a, b*], i.e. any value from this interval is represented in a memory as one value *x*.

As values are used in numerical operations it is necessary to ask, at least in the case of basic arithmetic operations, what is the influence to the precision?

Let as assume that we have two numbers *x* and *y* *x* = [a,b], *y* = [c,d].

The following interpretation of the basic arithmetic operations demonstrate how actual precision is defined.

- $x + y$ = [a + c, b + d]
- $x - y$ = [a - d, b - c]
- $x \times y$ = [min(ac, ad, bc, bd), max(ac, ad, bc, bd)]
- x / y = [min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)] if y ≠ 0

There are well known identities like

$$cos^2\alpha + cos^2\beta = 1$$

and

$$x^2 - y^2 = (x - y)(x + y)$$

However these identities are not valid if the floating point representation is used. For a computation of $x^2 - y^2$ it is better to use $(x - y)(x + y)$ due to better precision in evaluation, as if $|x| > |y|$ then $x^2 \gg y^2$ and therefore some last digits of the y^2 mantissa might be lost in the final subtraction.

Actually all statements like

- **if** <**float**> = <**float**> **then** ….
- **if** <**float**> ≠ <**float**> **then** ….

should not be allowed in programming languages or at least a warning message should be generated. Usually this problem is "solved" by constructions

if abs (x - y) < *epsilon* **then** … or

if abs (x) < *epsilon* **then** q:= y / x **else** ERROR ,

but nobody knows what is the proper value of *epsilon*.

Let us explore a little bit the numerical problems on very simple examples, now.

2.1 Quadratic Equation Solution

The quadratic equation is well known and used it is a part of many engineering problems solutions. Let us consider two formulations as follows [19]:

$$at^2 + bt + c = 0 \quad \text{resp.} \quad t^2 + pt + q = 0$$

The solution usually used is

$$t_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

resp.

$$t_{1,2} = \frac{-p \pm \sqrt{p^2 - 4q}}{2}$$

or if substitute $t = {}^1/_\tau$

$$\tau_{1,2} = \frac{2c}{-b \pm \sqrt{b^2 - 4ac}}$$

However in some cases the "standard" formula can lead to incorrect results due to a limited number precision. If $b^2 \gg 4ac$ then it is recommended to use the following formula

$$q = -(b + sign(b)\sqrt{b^2 - 4ac}\,)/2$$
$$t_1 = {}^{q}/_{a} \qquad t_2 = {}^{c}/_{a}$$

to get more reliable results.

It can be seen that even such a simple case might be quite sensitive to the numerical precision.

2.2 Function Value Computation

Computation of a function value is one of the basic common operations in engineering problems. However many programmers are not aware of the danger in the coding process. There seems to be two the most dangerous cases:

- division by a value close to zero, e.g. in an intersection computation of two nearly parallel lines
- addition or subtraction of two values with significantly different absolute value, e.g. recently mentioned $x^2 \pm y^2$.

As the result of this, the summation (repeated addition) result depends on the order of summation in general.

Let us explore one very interesting case [20] and some other interesting comments [1], [13], [18].

$$f(x,y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + x/(2y)$$

The question is, what is the value of the function if it is evaluated at $x = 77617$, $y = 33096$ if different floating point precision is used.

$f = 6.33835\ 10^{29}$ in single precision

$f = 1{,}1726039400532$ in double precision

$f = 1{,}172603940053178631858834904520183 8$ in extended precision

However even the result in the extended precision is incorrect and even the sign itself is incorrect. The correct! The correct result is "somewhere" in the interval of

$[-0{,}827396059946821368141165095479816\ 29\mathbf{2005},\ -0{,}827396059946821368141165095479816\ 29\mathbf{1986}]$

if approx. 40 digits were used [13]. Of course this function is constructed in a special way, but it demonstrate that

- simple increase of precision does not guarantee the correctness of the result
- roundoff error has significant influence to for a limited floating point computation.

Detailed analysis of this function can be found in [1] and the correct result is

$$f(x,y) = -2 + \frac{x}{2y} = \frac{54767}{66192}$$

Unfortunately precision of the numerical results are significantly influenced by compilers properties and options used, as the optimization of the code is not considering the numerical stability issues.

2.3 Addition and Computational Order

So far we have dealt with "complicated cases", usually seen as "not practical". Power series summation is one of the very practical and often used computations. Let us imagine simple examples of summation if single precision is used [52]:

$$\sum_{i=1}^{10^3} 10^{-3} = 0.999990701675415$$

or

$$\sum_{i=1}^{10^4} 10^{-4} = 1.000053524971008$$

It can be seen that in the both cases the result should be one. The correctness in summation is very important in power series computations, e.g.

$$\sum_{n=1}^{10^6} \frac{1}{n} = 14.357357$$

or if the reverse order is used

$$\sum_{n=10^6}^{1} \frac{1}{n} = 14.392651$$

It means that even for a small number of elements we do not obtain correct results.

2.4 Recursion

Recursion is very useful tool for finding a nice description of a problem solution, e.g. well known Tower of Hanoi, however if implemented directly it might causes some problems, like the stack overflow etc. The algorithm itself can be described as follows:

```
MOVE (A, C, n);
{    MOVE (A, B, n-1);
     MOVE (A, C, 1);
     MOVE (B, C, n-1)
}
# MOVE (from, to, number) #
```

This recursive elegant solution is simple to implement and only stack overflow can be expected; Iterative solution is known as well.

The recursive definition usually leads to two main searching strategies in implementation:

- depth first search
- breath first search

Those strategies are fundamental to artificial intelligence methods.

Let us explore recursive definition of the well known Ackermann function. The Ackerman function [44] is defined as follows:

$$A(m,n) = \begin{cases} n+1 & if\ m = 0 \\ A(m-1,1) & if\ M > 0\ and\ n = 0 \\ A\big(m-1, A(m,n-1)\big) & if\ m > 0\ and\ N > 0 \end{cases}$$

This function is simple, but the problem its computation as the value of the function grows very fast as

$$A(4,4) = 2^{2^{2^{65536}}} = 2^{2^{10^{197296}}}$$

As the computation is made in integers, no overflow is detected at all.

However engineering applications are more oriented to computation with numbers in floating point representation.

2.4 Continuous Fractions

There is one very interesting approach based on continuous fractions. It enables to represent even irrational numbers in some cases. The basic definition can be described as:

$$x = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_4 + \cfrac{a_3}{\dots}}}}$$

If $a_i \neq 1$ then it is the case of generalized continuous fractions. If $a_i = 1$ then π can be expressed as $\pi = [3; 7,15,1,292,1,1,1,2,1,3,1 \dots]$. As $\pi = 4\,arctan\,(1)$ then π can be expressed as [48]

$$\pi = \cfrac{4}{1 + \cfrac{1^2}{3 + \cfrac{2^2}{5 + \cfrac{3^2}{\dots}}}}$$

We can see that this number representation is quite different and detailed description can be found in [14].

We have presented some selected fundamental issues in numerical computations that have direct influence to results of numerical computation.

There is a significant question how today's computations are reliable and robust as we are using a continuous mathematical models, but using discrete systems for physical phenomena representation; number of digits for a number representation is limited. Only very careful coding with regard to numerical errors can prevent disaster situations and possible losses on humans.

2.4 Matrix Inversion

Matrix inversion is very often used in solution of engineering problems. However in many cases the matrix is ill conditioned and the results are not checked to the correctness of the solution. Many libraries available just return a matrix, which might be far from the matrix inverted we would expect without any message or warning message.

Let us assume a matrix inversion as

$$\boldsymbol{Ax} = \boldsymbol{b} \qquad\qquad \boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

and the Hilbert's Matrix

$$\boldsymbol{H}_{ij} = \frac{1}{i+j-1}$$

then the inversion of the matrix is known in the analytical form and can be expressed as

$$\boldsymbol{H}_{ij}^{-1} = (-1)^{i+j}(i+j-1) \binom{n+i-1}{n-j}\binom{n+j-1}{n-i}\binom{i+j-2}{i-1}^2$$

The inversion of the Hilbert's matrix can be used to evaluate algorithms or available numerical library for the stability and correctness of results delivered. Matrix inversion and a linear system of equations can be solved effectively without division operation if projective geometry is used [21], [22].

3 Numerical Disasters

There are famous examples of numerical disasters. When reading the original reports and followed comments and details one must be really surprised how simple errors occur and should be worried what could happen in complex problems solution. Let us shortly explore some "traditional" cases.

The following is a modified excerpt from public resources [45] - [47], [50], [51], [54].

3.1 Explosion of Ariane 5

An Ariane 5 rocket was launched by the European Space Agency (ESA) on June 4, 1996. The development cost over $7 billion. The rocket

exploded after lift-off in about 40 sec. Destroyed rocket and cargo were valued at $500 million. The cause of a failure was a software error in inertial reference system. From the CNN article:

"*The failure of the Ariane 501 was caused by the complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after lift-off). This loss of information was due to specification and design errors in the software of the inertial reference system.*

The internal SRI [Inertial Reference System] software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer."

The conversion from the floating point to the integer representation is very dangerous as it is not reported by an exception and stored value represents an existing number.

Figure 2
Courtesy of CNN

3.2 Patriot Missile Failure

The system was originally designed in mid-1960 for a short and flexible operation. There were several mishaps in the Patriot system failure. The system was actually running for more than 100 hours) and for intercepting cruise missiles running at MACH 2 speed and was used to intercept the Scud missile running at MACH 5. The computation of intercepting and hitting was based on time counting with 24 bits integers with the clock of 1/10[s] and speed computation in floats. The clock setting to 1/10[s] was a critical issue and not acceptable even for application in sport activities at that time. Unfortunately $1/10 = 1/2^4+1/2^5+1/2^8+1/2^9+1/2^{12}+....$ and therefore the error on 24 bits is about 0.000000095 and in 100 hours the error is 0.34. As the Scud flies at MACH 5, the error was actually 687[m] and the missile was out of the "range gate" area.

As a result of the fault assumptions, incorrect software design and irresponsible attitude of the army officials, 28 Americans were killed and over 100 other people injured in the Iraq's Scud missile attack in Dhahran, Saudi Arabia on February 25, 1991 according to the GAO report.

3.1 Offshore Platform Sinking

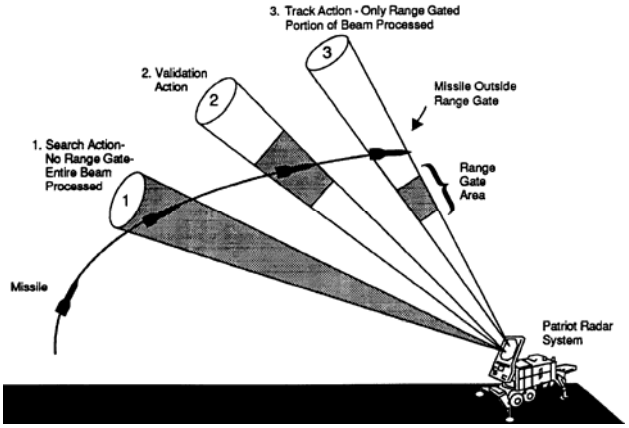


Figure 3
Courtesy of GAO report

Another well known example is the Sleipner offshore platform sinking. It should be noted that the top deck is about 57 000 tons, drilling and support equipments weight about 40 000 tons and the deck provides an accommodation for about 200 people.

Courtesy of SINTEF
Figure 4

The Sleipner platform structure was "optimized" using finite element system and the shear stresses were underestimated nearly by 50%. It led to serious cracks in the structure and leakage that the pumps were unable to cope with [51]. The sinking of the platform estimated cost is about $700 million.

All the above mentioned examples are well known; images and the text were used or adapted from the referenced resources.

We have presented above some basic facts on numerical precision and examples of some disasters. Many engineering problems are somehow connected with geometry and geometrical computations with respecting physical phenomena etc. The majority of computations are made in the Euclidean space representation and with the Cartesian coordinate system.

In the following we will show how the non-Euclidean representation, actually its projective extension, and the principle of duality can be used to solve some problems in a simple, robust and elegant ways.

4 Projective Space and Duality

The Euclidean space representation is used in today's computations using the floating point representation. Unfortunately the imprecision of the floating point computations is given by a number of mantissa digits that is limited. However the robustness of algorithms is more connected with the mathematical formulation and the actual implementation as well.

In many cases the Euclidean representation leads to unnecessary computations that even decrease the computational precision. The division operation is heavily used in engineering computations and it decreases the precision of computation significantly. There is a question if the division operation can be eliminated or at least postponed within the computational pipeline. In geometry, the projective representation is a way how things could be made simple, robust and easy to implement. Due to matrix-vector architecture additional speed can be expected as well.

4.1 Projective Space Representation

The projective space is actually an extension of the Euclidean space where no metrics is directly available. However it has several advantages, e.g. a point at infinity is well represented and can be used for computation.

The homogeneous coordinates are mostly introduced with the geometric transformations concept and used for the projective space representation. Many books and technical papers define mathematically how to make transformations from the homogeneous coordinates to the Euclidean coordinates and vice versa. However, geometrical interpretation is missing in nearly all publications. Therefore, the question is how to imagine the projective space P^2 and representations of elements.

Mutual transformations for the E^2 case are defined as:

$$X = x / w \quad Y = y /$$

where: $w \neq 0$, point $\boldsymbol{x} = [x, y: w]^T$ and $\boldsymbol{x} \in P^2$, $\boldsymbol{X} = [X, Y]^T$ and $\boldsymbol{X} \in E^3$.

Let us consider a situation at Fig.5.a. We can see that the point $\boldsymbol{X} \in E^2$ in the Euclidean space is actually a line p in the projective space P^2 passing the given point $\boldsymbol{X} \in E^2$ at the plane $w = 1$ (that is the Euclidean space actually) and the origin of the projective space P^2. It means that all the points $\boldsymbol{x} \in P^2$ of the line (excluding the origin at $[0, 0: 0]^T$) represent the same point in the Euclidean space. Similarly, transformation for the E^3 case is defined as:

$$X = x / w \quad Y = y / w \quad Z = z / w$$

where: $w \neq 0$, point $\boldsymbol{x} = [x, y, z: w]^T$ and $\boldsymbol{x} \in P^3$, $\boldsymbol{X} = [X, Y, Z]^T$ and $\boldsymbol{X} \in E^3$.

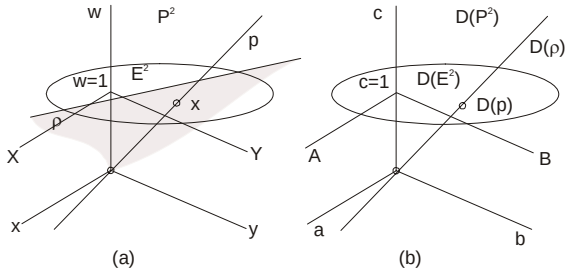


Figure 5
Euclidean, projective and dual space representations

Let us assume the Euclidean space E^2, see Fig.5.a. We actually use the projective space whenever we use the implicit representation for graphical elements. The Euclidean space E^2 is represented as a plane $w = 1$. For simplicity, let us consider a line p defined as:

$$aX + bY + c = 0$$

We can multiply it by $w \neq 0$ and we get:

$$awX + bwY + cw = 0 \text{ , i.e}$$

$$ax + by + cw = 0$$

i.e.

$$\boldsymbol{p}^T\boldsymbol{x} = 0$$

$$\boldsymbol{p} = [a, b: c]^T \qquad \boldsymbol{x} = [x, y: w]^T$$

It is actually a plane ρ in the projective space P^2 (excluding the point $[0, 0: 0]^T$, i.e. the origin) passing through the origin. The vector of coefficients $\boldsymbol{p}$ represents the line $p \in E^2$:

$$\boldsymbol{p} = [a, b: c]^T$$

Let us assume a dual representation in Fig.5.b. In the dual representation in which the point $[a, b: c]^T$ actually represents a line $D(p) \in D(E^2)$ given by the point $[a, b: c]^T$ and the origin of the dual space, see [29], [30], [38], [41], [42], [25], [27] for details on projective geometry.

It is necessary to note that any $\xi \neq 0$ can multiply the equation for a line without any effect to the geometry. It means that there are different vectors of coefficients $\boldsymbol{p}$ that represent the same line $p \in E^2$.

In the dual coordinate system, those points $[a, b: c]^T$ will form a line $D(p)$. We can project the line $D(p)$, e.g. to a plane with $c = 1$, and we get a point. It means that the line $p \in E^2$ is a point in the **dual representation** $D(p) \in D(E^2)$ and vice versa.

This a phenomenon of a principle of duality that can be used for derivation of some useful formula.

4.2 Principle of Duality

Let us consider an equation

$$\boldsymbol{p}^T\boldsymbol{x} = 0$$

From the mathematical notation we cannot distinguish whether $\boldsymbol{p}$ is a line and $\boldsymbol{x}$ is a point or vice versa in the case of P^2. It means that a point and a line are dual in the case of P^2, and a point and a plane are dual in the case of P^3.

The principle of duality in P^2 states that any theorem remains true when we interchange the words "point" and "line", "lie on" and "pass through", "join" and "intersection", "collinear" and "concurrent" and so on. Once the theorem has been established, the dual theorem is obtained as described above, see [12], [5], [6], [26], [27], [28], [35] for details.

In other words, the principle of duality says that in all theorems it is possible to substitute the term "point" by the term "line" and the term "line" by the term "point" etc. in E^2 and the given theorem stays valid. Similar duality is valid for E^3 as well, i.e. the terms "point" and "plane" are dual etc. This helps a lot to solve some geometrical problems [3], [4], [11], [15], [17].

4.2.1 E^2 Case

In the E^2 case, parameters of a line given by two points or an intersection point of two lines are computed very often. We will use the duality principle in which a point is dual to a line and vice versa.

In the first case, the solution is simple if the points are not in the homogeneous coordinates. If they are given in the homogeneous coordinates, the coordinates are converted to the Euclidean coordinates and then parameters of the line are computed.

In the second case, a linear system of equations of the degree two is usually solved and division is to be performed. It is necessary to note that any division operation decreases robustness of computation.

A new approach performing an appropriate computation in projective space is presented [31] - , [34]. It will allow us to avoid division operations.

Definition$_1$

The cross-product of two vectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \in E^2$, if given in the homogeneous coordinates, is defined as (if $w = 1$ the standard formula is obtained):

$$\mathbf{x}_1 \times \mathbf{x}_2 = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix}$$

where: $\boldsymbol{i} = [1,0:0]^T$, $\boldsymbol{j} = [0,1:0]^T$, $\boldsymbol{k} = [0,0:1]^T$

$$w_1 w_2 (\mathbf{X}_1 \times \mathbf{X}_2) = \mathbf{x}_1 \times \mathbf{x}_2 =$$

$$= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{vmatrix} = w_1 w_2 \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{x_1}{w_1} & \frac{y_1}{w_1} & 1 \\ \frac{x_2}{w_2} & \frac{y_2}{w_2} & 1 \end{vmatrix} =$$

$$w_1 w_2 \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \end{vmatrix}$$

Theorem$_1$

Let two points $\boldsymbol{x}_1, \boldsymbol{x}_2 \in E^2$ be given in the projective space. Then a line $p \in E^2$ defined by those two points is determined as a cross-product:

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$$

where: $\boldsymbol{p} = [a, b: c]^T$ and $\boldsymbol{x}_i = [x_i, y_i: w_i]^T$

Proof$_1$

Let the line $p \in E^2$ is defined as:

$$ax + by + c = 0$$

The end-points must satisfy $\mathbf{x}_1^T\mathbf{p} = 0$ and $\mathbf{x}_2^T\mathbf{p} = 0$, i.e.

$$\begin{bmatrix} x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

It results to a standard:

$$a = \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \quad b = -\begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \quad c = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}$$

and therefore the cross-product defines the line p, i.e.

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$$

□

Note: This is valid also generally for all the cases when $w \neq 0$. The proof is left to a reader.

Now we can apply the principle of duality directly.

Theorem$_2$

Let two lines $\boldsymbol{p}_1, \boldsymbol{p}_2 \in E^2$ be given in the projective space. Then a point $\boldsymbol{x}$ defined as an intersection of those two lines is determined as a cross product:

$$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2.$$

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

Proof$_2$

This is a direct consequence of the principle of duality application.

$$\mathbf{x} = \mathbf{p}_1 \times \mathbf{p}_2 = \begin{vmatrix} x & y & w \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{vmatrix}$$

where: $\boldsymbol{x} = [x, y: w]^T$

□

These two theorems are very important as they enable us to handle some problems defined in the homogeneous coordinates directly and make computations quite efficient as we can postpone the division operation.

The direct impact of these two theorems is that it is very easy to compute a line given by two points in E^2 and an intersection point of two lines in E^2 as well. The presented approach is convenient if matrix-vector operations are supported in hardware, especially for GPU applications.

Note that we do not need to solve linear system of equations to find the intersection point of two lines and if the result can remain in the homogeneous coordinates, no division operation is needed.

Of course, there is a question, how to handle the E^3 cases.

4.2.2 E^3 Case

The E^3 case is a little bit complicated as the projective geometry and duality offer more possibilities, but generally a point is dual to a plane and vice versa. So let us explore how to find:

- a plane defined by three points given in the homogeneous coordinates,
- an intersection point of three planes.

To find a plane is simple if points are converted to the Euclidean coordinates. It requires use of the division operation and therefore robustness is decreased in general.

Let us explore the extension possibility of the E^2 cases, as discussed above, to the E^3 case.

Definition$_2$

The cross-product of three vectors $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $\boldsymbol{x}_3$ is defined as:

$$\mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3 = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{vmatrix}$$

where: $\boldsymbol{i} = [1,0,0:0]^T$, $\boldsymbol{j} = [0,1,0:0]^T$, $\boldsymbol{k} = [0,0,1:0]^T$, $\boldsymbol{l} = [0,0,0:1]^T$ and $\boldsymbol{x}_i = [x_i, y_i, z_i: w_i]^T$

Theorem$_3$

Let three points $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ be given in the projective space. Then a plane $\boldsymbol{\rho} \in E^3$ defined by those three points is determined as:

$$\boldsymbol{\rho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3$$

Proof$_3$

Let the plane $\boldsymbol{\rho} \in E^3$ be defined as:

$$aX + bY + cZ + d = 0$$

or

$$ax + by + cz + dw = 0$$

It can be seen that:

$$a = \begin{vmatrix} y_1 & z_1 & w_1 \\ y_2 & z_2 & w_2 \\ y_3 & z_3 & w_3 \end{vmatrix} \quad b = -\begin{vmatrix} x_1 & z_1 & w_1 \\ x_2 & z_2 & w_2 \\ x_3 & z_3 & w_3 \end{vmatrix}$$

$$c = \begin{vmatrix} x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \\ x_3 & y_3 & w_3 \end{vmatrix} \quad d = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

that is the cross-product that defines a plane ρ if three points are given and therefore:

$$\boldsymbol{\rho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3$$

□

Note: The proof of the standard formula, i.e. for the case $w = 1$, is left to a reader.

As a point is dual to a plane, a plane is dual to a point we can use the principle of duality directly, now.

Theorem$_4$
Let three planes $\boldsymbol{\rho}_1, \boldsymbol{\rho}_2$ and $\boldsymbol{\rho}_3$ be given in the projective space. Then a point $\boldsymbol{x}$, which is defined as the intersection point of those three planes, is determined as:

$$\boldsymbol{x} = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3$$

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

Proof$_4$
This is a direct consequence of the principle of duality application:

$$\mathbf{x} = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3 = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{vmatrix}$$

where: $\boldsymbol{x} = [\, x, y, z: w]^T$ and $\boldsymbol{\rho}_i = [a_i, b_i, c_i: d_i]^T$
□
These two theorems are very important as they enable us to handle some problems defined in the homogeneous coordinates efficiently and make computations quite efficient. Even more, if an input is in the Euclidean or the homogeneous coordinates and output can be in the homogeneous coordinates, no division is needed. In the case of two parallel lines, the homogeneous coordinate $w=0$. It means that we have robust computation of an intersection point.

The direct impact of these two theorems is that it is very easy to compute a plane in the E^3 given by three points in the E^3 and compute an intersection point determined as an intersection of three planes in the E^3 and only implementation of one routine is needed. Of course, there is a question, how to handle lines in the E^3 or P^3 cases.

The above mentioned formulae using the projective notation are not well known in general and the authors present explicit formulae for the Euclidean coordinates, i.e. for $w = 1$.

4.2.3 Line in E^3 defined parametrically

Let us consider a little bit more difficult problems formulated as follows:

- determine a line $q \in E^3$ if given by two points $\boldsymbol{x}_i$,
- determine a line $q \in E^3$ if given by two planes $\boldsymbol{\rho}_i$.

if a parametric form is required.

These problem formulations seem to be trivial problems if $w_i = 1$ or the division operations are permitted. In the case of $w_i \neq 1$ situation is more complicated as the points can be converted from the projective space to the Euclidean space using a 3 division operations per a point, i.e.

2 points × 3 divisions = 6 divisions

However this is not necessarily needed in many algorithms actually as we need only ordering information, i.e. if the nearest intersection of a line and several planes is computed. It means that we need the order of intersection. In this case we can use a linear interpolation with non-linear monotonical parameterization. Let us imagine two points

$$\boldsymbol{x}_1 = [x_1, y_1, z_1: w_1]^T \qquad \boldsymbol{x}_2 = [x_2, y_2, z_2: w_2]^T$$

Then the line in a parametric form can be defined as

$$\boldsymbol{x}(t) = \boldsymbol{x}_1 + (\boldsymbol{x}_2 - \boldsymbol{x}_1)\, t = \boldsymbol{x}_1 + \boldsymbol{s}\, t$$

where the vector $\boldsymbol{s}$ is computed as a difference in the projective space, i.e. without transformation to the Euclidean space. Nevertheless monotonic parameterization is required in many applications and points are given in the projective space.

Let us consider a two points given in the homogeneous coordinates.

On the other hand, a classic rule for robustness is to "postpone division operation to the last moment possible". Even if division is permitted, the 2nd case seems to be more difficult not only from the robustness point of view as the line is considered as an intersection of two planes, i.e. a common solution of their implicit equations.

We will derive a new method for the computation of a line in the E^3 for those two possible cases without use of division directly in the projective space.

The Plücker coordinates will be used as they can help us to formalize and resolve this problem efficiently.

4.3 Plücker Coordinates

The formulae presented above enable us to handle points and planes in E^3. Nevertheless, it is necessary to handle lines in the E^3 in the parametric form using the homogeneous coordinates as well and avoid the division operations, too.

A parametric form for a line given by two points in the Euclidean space is given as:

$$\boldsymbol{X}(t) = \boldsymbol{X}_1 + (\boldsymbol{X}_2 - \boldsymbol{X}_1)\, t$$

where: t is a parameter $t \in (-\infty, \infty)$.

This is straightforward for the Euclidean coordinates and for the homogeneous coordinates if the division operation is permitted. It is necessary to represent a position and a direction. The question is how to make it directly in the projective space using the homogeneous coordinates.

Therefore, in the following the Plücker coordinates will be introduced to resolve the case. Another approach using the Grassmann coordinate system can be found in [6].

Let us consider two points in the homogeneous coordinates:

$$\boldsymbol{x}_1 = [x_1, y_1, z: w_1]^{\mathrm{T}} \qquad \boldsymbol{x}_2 = [x_2, y_2, z_2: w_2]^{\mathrm{T}}$$

The Plücker coordinates l_{ij} are defined as follows:

$$l_{41} = w_1x_2 - w_2x_1 \qquad l_{23} = y_1z_2 - y_2z_1$$
$$l_{42} = w_1y_2 - w_2y_1 \qquad l_{31} = z_1x_2 - z_2x_1$$
$$l_{43} = w_1z_2 - w_2z_1 \qquad l_{12} = x_1y_2 - x_2y_1$$

It is possible to express the Plücker coordinates as

$$l_{ij} = \mathbf{x}_1^{(i)}\mathbf{x}_2^{(j)} - \mathbf{x}_2^{(i)}\mathbf{x}_1^{(j)}$$

alternatively, as an anti-symmetric matrix $\boldsymbol{L}$:

$$\mathbf{L} = \mathbf{x}_1\mathbf{x}_2^T - \mathbf{x}_2\mathbf{x}_1^T$$

where: $l_{ij} = -\, l_{ji}$ and $l_{ii} = 0$.

Let us define two vectors $\boldsymbol{\omega}$ and $\mathbf{v}$ as:

$$\boldsymbol{\omega} = [l_{41}\,,\, l_{42}\,,\, l_{43}\,]^{\mathrm{T}} \qquad \boldsymbol{v} = [l_{23}\,,\, l_{31}\,,\, l_{12}\,]^{\mathrm{T}}$$

It means that $\boldsymbol{\omega}$ represents the "directional vector", while $\mathbf{v}$ represents the "positional vector". It can be seen that for the Euclidean space ($w = 1$) we get:

$$\boldsymbol{X}_2 - \boldsymbol{X}_1 = \boldsymbol{\omega} \qquad \boldsymbol{X}_1 \times \boldsymbol{X}_2 = \boldsymbol{v}$$

where: $\boldsymbol{X}_i = [x_i, y_i, z_i]^{\mathrm{T}}/w_i$ are points in the Euclidean coordinates.

For the general case $w_i \neq 1$ when $\boldsymbol{x}_i$ are not ideal points, i.e. $w_i \neq 0$ we get:

$$\mathbf{X}_2 - \mathbf{X}_1 = \left(\frac{x_2}{w_2} - \frac{x_1}{w_1}, \frac{y_2}{w_2} - \frac{y_1}{w_1}, \frac{z_2}{w_2} - \frac{z_1}{w_1}\right)$$

It can be seen that for the projective space, vectors $\boldsymbol{\omega}$ and $\boldsymbol{v}$ can be expressed as:

$$\boldsymbol{\omega} = w_2w_1\left(\mathbf{X}_2 - \mathbf{X}_1\right) =$$
$$\left(x_2w_1 - x_1w_2\,,\, y_2w_1 - y_1w_2\,,\, z_2w_1 - z_1w_2\right)$$
$$= \left(l_{41}, l_{42}, l_{43}\right)$$

and

$$\mathbf{v} = w_2w_1\left(\mathbf{X}_1 \times \mathbf{X}_2\right) =$$
$$\left(y_1z_2 - y_2z_1, z_1x_2 - z_2x_1, x_1y_2 - x_2y_1\right) =$$
$$\left(l_{23}, l_{31}, l_{12}\right)$$

The equations above show the relation between vectors $\boldsymbol{\omega}$ and $\boldsymbol{v}$ and the Plücker coordinates l_{ij}. In 1871 Klein derived that $\boldsymbol{\omega}^{\mathrm{T}}\,\boldsymbol{v} = 0$ [12], i.e. in the Plücker coordinates:

$$l_{23} * l_{41} + l_{31} * l_{42} + l_{12} * l_{43} = 0$$

This is a homogeneous equation of degree 2 and therefore the solution lies on a 4-dimensional quadratic hyper-surface. If q is a point on a line $\boldsymbol{q}(t) = \boldsymbol{q}_1 + \boldsymbol{\omega}\, t$ given by the Plücker coordinates, it must satisfy equation:

$$\boldsymbol{\omega} \times \mathbf{q} = \mathbf{v}$$

Let $\boldsymbol{X}_2 - \boldsymbol{X}_1 = \boldsymbol{\omega}$ and $\boldsymbol{X}_1 \times \boldsymbol{X}_2 = \boldsymbol{v}$. A point on the line $\boldsymbol{q}(t) = \boldsymbol{q}_1 + \boldsymbol{\omega}\, t$ is defined as:

$$\mathbf{q}(t) = \frac{\mathbf{v} \times \boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2} + \boldsymbol{\omega} t$$

Please, see Appendix C for derivation of this formula. It should be noted that for $t = 0$ we do not get the point $\boldsymbol{X}_1$. If $\|\boldsymbol{\omega}\| = 0$ the given points are equal.

The equation defines a line $\boldsymbol{q}(t)$ in the E^3 by two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ given in the homogeneous coordinates. Of course, we can avoid the division operation easily using homogeneous notation for a scalar value $\tilde{\boldsymbol{q}}(t)$, as follows:

$$\tilde{\boldsymbol{q}}(t) = \begin{bmatrix} \boldsymbol{v} \times \boldsymbol{\omega} + t\boldsymbol{\omega}\|\boldsymbol{\omega}\|^2 \\ \|\boldsymbol{\omega}\|^2 \end{bmatrix}$$

and the resulting line is defined directly in the projective space P^3.

Let us imagine that we have to solve the second problem, i.e. a line defined as an intersection of two given planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ in the Euclidean space:

$$\boldsymbol{\rho}_1 = [a_1, b_1, c_1, d_1]^{\mathrm{T}} \qquad \boldsymbol{\rho}_2 = [a_2, b_2, c_2, d_2]^{\mathrm{T}}$$

It is well known that the directional vector $\boldsymbol{s}$ of the line is given by those two planes as a ratio:

$$s_x : s_y : s_z = \begin{vmatrix} b_1 & c_1 \\ b_2 & c_2 \end{vmatrix} : \begin{vmatrix} c_1 & a_1 \\ c_2 & a_2 \end{vmatrix} : \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}$$

that is actually the ratio $l_{23} : l_{31} : l_{12}$ if the principle of duality is used, i.e. vector of $[a_i, b_i, c_i, d_i]^{\mathrm{T}}$ instead of $[x_i, y_i, z_i, w_i]^{\mathrm{T}}$ is used, and it defines the vector $\boldsymbol{v}$ instead of $\boldsymbol{\omega}$.

Now we can apply the principle of duality as we can interchange the terms "point" and "plane" and exchange $\boldsymbol{v}$ and $\boldsymbol{\omega}$ in the Eq.34 and we get:

$$\mathbf{q}(t)=\frac{\boldsymbol{\omega}\times\mathbf{v}}{\|\mathbf{v}\|^2}+\mathbf{v}t$$

and similarly to the Eq.35, the formula for the line in the *homogeneous coordinates is given as:*

$$\tilde{\mathbf{q}}(t)=\begin{bmatrix}\boldsymbol{\omega}\times\mathbf{v}+t\mathbf{v}\|\mathbf{v}\|^2\\ \|\mathbf{v}\|^2\end{bmatrix}$$

If $\|\mathbf{v}\|=0$ *then the given planes are* parallel.

It means that we have obtained the known formula for an intersection of two planes $\boldsymbol{\rho}_1, \boldsymbol{\rho}_2$ in the Euclidean coordinates, see [7]:

$$\mathbf{q}(t)=\mathbf{q}_0+\mathbf{n}_3\,t$$

where: $\mathbf{n}_3=\mathbf{n}_1\times\mathbf{n}_2$, $\boldsymbol{q}_0=[X_0, Y_0, Z_0]^{\mathrm{T}}$ and planes

$$\boldsymbol{\rho}_1:\mathbf{n}_1{}^T\mathbf{x}+d_1=0 \qquad \boldsymbol{\rho}_2:\mathbf{n}_2{}^T\mathbf{x}+d_2=0$$

The intersection point $\boldsymbol{X}_0$ of three planes in the Euclidean coordinates is defined as:

$$x_0=\frac{d_2\begin{vmatrix}b_1 & c_1\\ b_3 & c_3\end{vmatrix}-d_1\begin{vmatrix}b_2 & c_2\\ b_3 & c_3\end{vmatrix}}{DET}$$

$$y_0=\frac{d_2\begin{vmatrix}a_3 & c_3\\ a_1 & c_1\end{vmatrix}-d_1\begin{vmatrix}a_3 & c_3\\ a_2 & c_2\end{vmatrix}}{DET}$$

$$z_0=\frac{d_2\begin{vmatrix}a_1 & b_1\\ a_3 & b_3\end{vmatrix}-d_1\begin{vmatrix}a_2 & b_2\\ a_3 & b_3\end{vmatrix}}{DET}$$

$$DET=\begin{vmatrix}a_1 & b_1 & c_1\\ a_2 & b_2 & c_2\\ a_3 & b_3 & c_3\end{vmatrix}$$

If a line is defined by two points and $\|\boldsymbol{\omega}\|=1$, i.e. the directional vector is normalized, we get the standard formula and the line is simply determined as:

$$\mathbf{q}(t)=\mathbf{v}\times\boldsymbol{\omega}+\boldsymbol{\omega}t$$

As a point is dual to a plane we can directly write an equation for an intersection of two lines.

In the case of a line defined by two planes and $\|\mathbf{v}\|=1$, i.e. the positional vector is normalized, we get the line simply determined as:

$$\mathbf{q}(t)=\boldsymbol{\omega}\times\mathbf{v}+\mathbf{v}t$$

Those formulae are well known if the Euclidean coordinates are used.

Note:

It is possible to define vectors $\boldsymbol{v}$ and $\boldsymbol{\omega}$ for the plane intersection case as $\boldsymbol{v}=[l_{41}, l_{42}, l_{43}]^{\mathrm{T}}$ and $\boldsymbol{\omega}=[l_{23}, l_{31}, l_{12}]^{\mathrm{T}}$, i.e. with swapped Plücker vectors, and have the same equation for the line $\boldsymbol{q(t)}$ but the symbols would have different interpretation – that is the reason, why the priority was given to different notation for those two cases.

However, an intersection of two planes is the case very often solved in computer graphics and vision. Unfortunately in many cases available solutions are not robust or formula are neither simple, like above, nor convenient for GPU use.

In the following a new formulation of two plane intersection is presented and if the projective space is used for formulation, the solution is quite simple.

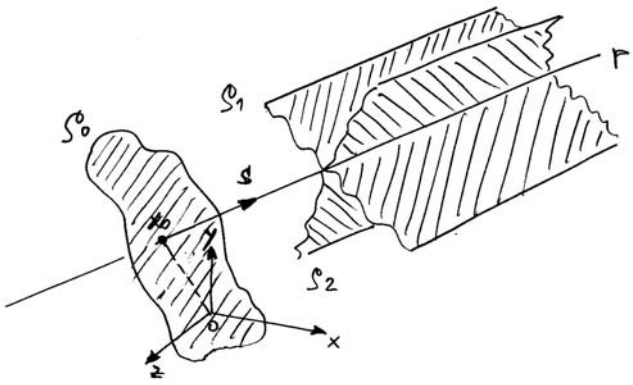


Figure 6. Intersection of two planes

If the projective space is used, the solution is quite simple. Let us consider two planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_1$ given as

$$\boldsymbol{\rho}_1=[a_1,b_1,c_1{:}\,d_1]^T \qquad \boldsymbol{\rho}_2=[a_2,b_2,c_2{:}\,d_2]^T$$

It means that normal vectors of those planes are

$$\boldsymbol{n}_1=[a_1,b_1,c_1]^T \qquad \boldsymbol{n}_2=[a_2,b_2,c_2]^T$$

It is obvious that a directional vector of a line is determined as an intersection of two planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_1$ given as

$$\boldsymbol{s}=\boldsymbol{n}_1\times\boldsymbol{n}_2$$

However, the "starting" point $\boldsymbol{x}_0$ of the line is determined in quite complicated ways, sometimes even not robustly enough and based on a user choice of some value, or proposes solution of a system of linear equations leading to a standard formula given above.

The formula is quite "horrible" one and for students not acceptable as it is too complex and they do not see from the formula comes from.

However, there is a quite simple geometrical explanation and solution. So the first question is how to find the "starting" point $\boldsymbol{x}_0$ of the line $\boldsymbol{p}$ given by two planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$. If a robust solution is required a user should be prevented from a selection of some "parameters".

Let us imagine that there exists a plane $\boldsymbol{\rho}_0$, whose normal vector is given as $\boldsymbol{s} = \boldsymbol{n}_1 \times \boldsymbol{n}_2$. It means that its position needs to be "fixed" in the space. As there is no other requirement on this plane, we can "fix" it so it passes through the origin of the coordinate system, i.e. the plane $\boldsymbol{\rho}_0$ is given as

$$\boldsymbol{\rho}_0 = [a_0, b_0, c_0 : 0]^T$$

and the line $\boldsymbol{p}$ is orthogonal to the plane $\boldsymbol{\rho}_0$ – resulting into a robust geometric position.

Now, the intersection point of those three planes is the point $\boldsymbol{x}_0$ we are looking for. Coordinates of the point $\boldsymbol{x}_0$ are determined by generalized cross-product as

$$\boldsymbol{x}_0 = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_0$$

It is obvious that the point $\boldsymbol{x}_0$ is also the closest point on the line to the origin, too. As this formula is very compact and it is suitable for GPU application. Appendix A presents the extended cross-product GPU implementation.

From the formulation presented above, it can be seen that it is not only very simple, easy to understand and remember, but also easy to implement as well. As a result, the Plücker coordinates formulation of this problem solution is not needed when looking for such properties.

4.2.1 Geometric Transformations with Duals

Geometric transformations in computer graphics and computer vision are mostly based on transformations of points. Nevertheless in many cases we have a line given by two points in E^2 or a plane given by three points in E^3. The question is how the line or the plane will change if a geometric transformation is applied on those points.

We need to determine a transformation matrix $\boldsymbol{Q}$ for a transformed line p', if geometric transformation $\boldsymbol{T}$ is applied on points defining the line p without a need to re-compute the coefficients of the line from the transformed points.

$$\boldsymbol{p}' = \boldsymbol{Q}(\boldsymbol{x}_1 \times \boldsymbol{x}_2) = (\boldsymbol{T}\boldsymbol{x}_1) \times (\boldsymbol{T}\boldsymbol{x}_2) = \frac{(\boldsymbol{T}^{-1})^T(\boldsymbol{x}_1 \times \boldsymbol{x}_2)}{det(\boldsymbol{T})}$$

it means that $\boldsymbol{Q} = (\boldsymbol{T}^{-1})^T / det(\boldsymbol{T})$.

For the standard geometric transformations rotation and translation $\det(\boldsymbol{T}) = 1$ the matrix $\boldsymbol{Q}$ is simple. Nevertheless the matrix $\boldsymbol{Q}$ can be determined for a general transformation. It should be noted that e.g. a rotation can be "rewritten" in the projective notation as

$$\boldsymbol{x}' = \boldsymbol{R}(\varphi)\,\boldsymbol{x} = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}$$

$$\boldsymbol{x}' = \boldsymbol{R}'(\varphi)\,\boldsymbol{x} \cong \begin{bmatrix} a & -b & 0 \\ b & a & 0 \\ 0 & 0 & c \end{bmatrix}$$

where: $\cos(\varphi) = a/c$ and $\sin(\varphi) = b/c$. In this case $\det(\boldsymbol{T}) \neq 1$ of course. As the specification is in the projective space, we can use $\boldsymbol{Q} = (\boldsymbol{T}^{-1})^T$ for line and plane transformations and save the division operation.

It can be seen that for the case of E^3 a similar approach can be taken as well.

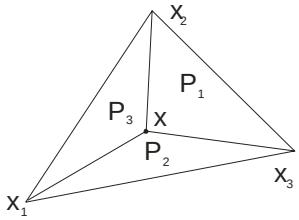
As a result of this approach is that we can easily solve a problem: Given a line p and a geometric transformation $\boldsymbol{T}$ in the projective space. How the coefficients of a line are changed? Similarly for a plane in E^3 and dual problems a solution is simple.

5 Barycentric coordinates

Barycentric coordinates are very often used in computer graphics. Usually a system of linear equations has to be solved. If the points forming a simplex are given in the projective space, solution requires use of division operation. However the barycentric coordinates can be solved without division operation if projective space and generalized cross-product are used.

5.1 Euclidean Barycentric Coordinates

In computer graphics Euclidean or homogeneous coordinates are widely used as well as parametric formulations, e.g. triangles, parametric patches etc. The barycentric coordinates have many useful and interesting properties [24].



Barycentric coordinates in E^2
Figure 7

Let us consider a triangle with vertices $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, see Fig.7. The position of any point $\mathbf{X} \in E^2$ can be expressed as

$$a_1 X_1 + a_2 X_2 + a_3 X_3 = X$$

$$a_1Y_1 + a_2Y_2 + a_3Y_3 = Y$$

If we add an additional condition

$$a_1 + a_2 + a_3 = 1$$

we get a system of linear equations. The coefficients a_i are called barycentric coordinates of the point **X**. The point **X** is inside the triangle **if and only if** $0 \le a_i \le 1$, i = 1,…,3. It is useful to know that

$$a_i = \frac{P_i}{P} \quad \text{i} = 1,...,3$$

where: P is the area of the given triangle and P_i is the area of the i-th subtriangle.

Note: The barycentric coordinates can easily be converted into the usual parametric form. It can be seen that $a_1 = 1 - a_2 - a_3$. Substituting this we obtain

$$(1 - a_2 - a_3)X_1 + a_2X_2 + a_3X_3 = X$$

i.e.

$$X_1 + a_2(X_2 - X_1) + a_3(X_3 - X_1) = X$$

and finally we get

$$a_2(X_2 - X_1) + a_3(X_3 - X_1) = X - X_1$$

It is the standard formula usually used. Similarly, it may be used for other coordinates.

Now a system of linear equations has to be solved, i.e.

$$\mathbf{A\alpha} = \boldsymbol{\beta}$$

where: $\boldsymbol{\alpha} = [a_1, a_2, a_3]^T$, $\boldsymbol{\beta} = [X, Y, 1]^T$ and

$$\mathbf{A} = \begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix}$$

and division operations must be used to solve this linear system of equations. In some cases, especially when the triangles are very thin, there might be a severe problem with the stability of the solution.

The non-homogeneous system of linear equations $\mathbf{A\alpha} = \boldsymbol{\beta}$ can be transformed into a homogeneous linear system

$$b_1X_1 + b_2X_2 + b_3X_3 + b_4X = 0$$
$$b_1Y_1 + b_2Y_2 + b_3Y_3 + b_4Y = 0$$
$$b_1 + b_2 + b_3 + b_4 = 0$$

where: $b_4 \neq 0$ and $b_i = -a_ib_4 \quad i = 1,...,3$.

Rewriting this system in a matrix form, we get

$$\begin{bmatrix} X_1 & X_2 & X_3 & X \\ Y_1 & Y_2 & Y_3 & Y \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \mathbf{0}$$

or in the matrix form

$$\mathbf{B\,b} = \mathbf{0} \qquad \textit{or} \qquad [\mathbf{A} \mid \mathbf{X}][\mathbf{b}] = \mathbf{0}$$

where: $\mathbf{b} = [b_1, b_2, b_3, b_4]^T$, $\mathbf{X} = [X, Y, 1]^T$,

$$\mathbf{A} = \begin{bmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = [\mathbf{A} \mid \mathbf{X}]$$

In another way, we are looking for a vector $\boldsymbol{\tau}$, see eq.4, that satisfies the condition

$$\boldsymbol{\tau}^T\mathbf{b} = 0$$

where: $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3, \tau_4]^T$

This equation can be expressed using the determinant form as:

$$\det \begin{vmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 \\ X_1 & X_2 & X_3 & X \\ Y_1 & Y_2 & Y_3 & Y \\ 1 & 1 & 1 & 1 \end{vmatrix} = 0$$

It is obvious that it can be formally written as:

$$\mathbf{b} = \boldsymbol{\xi} \times \boldsymbol{\eta} \times \mathbf{w}$$

where: $\mathbf{b} = [b_1, b_2, b_3, b_4]^T$ $\boldsymbol{\xi} = [X_1, X_2, X_3, X]^T$

$\boldsymbol{\eta} = [Y_1, Y_2, Y_3, Y]^T$ $\mathbf{w} = [1,1,1,1]^T$

and the barycentric coordinates of the point $\mathbf{X}$ are given as $a_1 = -\frac{b_1}{b_4}$, $a_2 = -\frac{b_2}{b_4}$, $a_3 = -\frac{b_3}{b_4}$

We can use the Plücker coordinates notation and write

$$a_i = (-b_i : b_4) \quad i = 1,...,3.$$

If $b_4 = 0$, the triangle is degenerated to a line segment or to a point, i.e. it is a singular case, which can be correctly detected.

The given point $\mathbf{X}$ is inside the given **triangle if and only if** $0 \le a_i \le 1$, i = 1,…,3. This condition is a little bit more complicated for the homogeneous representation and can be expressed by a sequence

if $b_4 > 0$ **then** $0 \le -b_i \le b_4$

else $b_4 \le -b_i \le 0 \quad i = 1,...,3$

This is a very important result as it means that **we do not need the division operation** for testing whether the given point $\mathbf{X}$ is inside the given triangle!

In many applications, the vertices of the given triangle and the given point $\mathbf{X}$ can be given in homogeneous coordinates. Let us explore how the barycentric coordinates could be computed in this case.

The linear system of equations for the barycentric coordinates can be rewritten as:

$$a_1\frac{x_1}{w_1}+a_2\frac{x_2}{w_2}+a_3\frac{x_3}{w_3}=\frac{x}{w}$$
$$a_1\frac{y_1}{w_1}+a_2\frac{y_2}{w_2}+a_3\frac{y_3}{w_3}=\frac{y}{w}$$
$$a_1+a_2+a_3=1$$

where: $\mathbf{x}_i=\left[x_i,y_i:w_i\right]^T$ represents the i-th vertex triangle in the homogeneous coordinates and $\mathbf{x}=\left[x,y,w\right]^T$ is the given point in the homogeneous coordinates.

We can multiply the linear system by $w\neq 0$, $w_i\neq 0 \quad i=1,...,3$ and substitute:

$$b_1=-a_1w_2w_3w \qquad b_2=-a_2w_1w_3w$$
$$b_3=-a_3w_1w_2w \qquad b_4=w_1w_2w_3w$$

Thus we get:

$$b_1x_1+b_2x_2+b_3x_3+b_4x=0$$
$$b_1y_1+b_2y_2+b_3y_3+b_4y=0$$
$$b_1w_1+b_2w_2+b_3w_3+b_4w=0$$

and in the matrix notation:

$$\begin{bmatrix}x_1 & x_2 & x_3 & x\\ y_1 & y_2 & y_3 & y\\ w_1 & w_2 & w_3 & w\end{bmatrix}\begin{bmatrix}b_1\\ b_2\\ b_3\\ b_4\end{bmatrix}=\mathbf{0}$$

We are looking for a vector $\boldsymbol{\tau}$ that satisfies the following equation:

$$\boldsymbol{\tau}^T\mathbf{b}=0$$

where: the vector $\boldsymbol{\tau}$ is defined as $\boldsymbol{\tau}=\left[\tau_1,\tau_2,\tau_3:\tau_4\right]^T$

Then the solution is defined as

$$\det\begin{vmatrix}\tau_1 & \tau_2 & \tau_3 & \tau_4\\ x_1 & x_2 & x_3 & x\\ y_1 & y_2 & y_3 & y\\ w_1 & w_2 & w_3 & w\end{vmatrix}=0$$

and we can formally write

$$\mathbf{b}=\boldsymbol{\xi}\times\boldsymbol{\eta}\times\mathbf{w}$$

where: $\mathbf{b}=\left[b_1,b_2,b_3,b_4\right]^T \quad \boldsymbol{\xi}=\left[x_1,x_2,x_3,x\ \right]^T$

$\boldsymbol{\eta}=\left[y_1,y_2,y_3,y\ \right]^T \qquad \mathbf{w}=\left[w_1,w_2,w_3,w\ \right]^T$

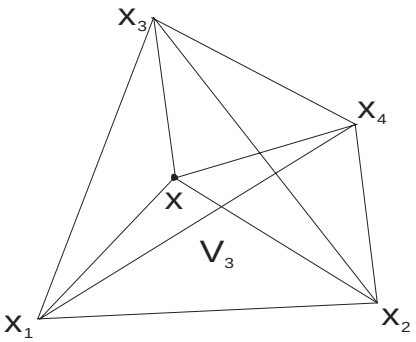
Of course, the conditions in the case that the point is inside the given triangle are slightly more complex, and the condition $0\le a_i\le 1 \quad i=1,...,3$ can be expressed by the following criteria:

$$0\le\left(-b_1:w_2w_3w\ \right)\le 1$$
$$0\le\left(-b_2:w_1w_3w\ \right)\le 1$$
$$0\le\left(-b_3:w_1w_2w\ \right)\le 1$$

This means that the barycentric coordinates can be computed **without using the division operation** even if the vertices of the given triangle and the point **x** are given in homogeneous coordinates. Therefore the approach presented here is more robust than the direct computation, i.e. normalizing the vertices and point coordinates into Euclidean coordinates and standard barycentric coordinates computation. In addition, the test if a point is inside the given triangle is consequently more robust.

Of course, there is a natural question: is it possible to extend the above mentioned approach to the E^3 case?

Let us consider the E^3 case, where the "point in a tetrahedron" test is similar to the "point in a triangle" test in E^2, see Fig.7.



Barycentric coordinates in E^3

Figure 7

It can be seen that the barycentric coordinates are given as

$$a_1X_1+a_2X_2+a_3X_3+a_4X_4=X$$
$$a_1Y_1+a_2Y_2+a_3Y_3+a_4Y_4=Y$$
$$a_1Z_1+a_2Z_2+a_3Z_3+a_4Z_4=Z$$
$$a_1+a_2+a_3+a_4=1$$

It is useful to know that

$$a_i=\frac{V_i}{V} \quad \text{i} = 1,...,3$$

where: V is the volume of the given tetrahedron and V_i is the volume of the i-th sub-tetrahedron.

The non-homogeneous system of linear equations can be transformed into a homogeneous linear system of equations

$$b_1X_1+b_2X_2+b_3X_3+b_4X_4+b_5X=0$$
$$b_1Y_1+b_2Y_2+b_3Y_3+b_4Y_4+b_5Y=0$$
$$b_1Z_1+b_2Z_2+b_3Z_3+b_4Z_4+b_5Z=0$$
$$b_1+b_2+b_3+b_4+b_5=0$$

where: $b_5\neq 0$ and $b_i=-a_ib_5 \quad i=1,...,4$

Rewriting this system in matrix form, we get

$$\begin{bmatrix} X_1 & X_2 & X_3 & X_4 & X \\ Y_1 & Y_2 & Y_3 & Y_4 & Y \\ Z_1 & Z_2 & Z_3 & Z_4 & Z \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \mathbf{0}$$

i.e.

$$\mathbf{B}\,\mathbf{b} = \mathbf{0} \qquad or \qquad [\mathbf{A} \mid \mathbf{X}][\mathbf{b}] = \mathbf{0}$$

where: $\mathbf{b} = [b_1, b_2, b_3, b_4 : b_5]^T$, $\mathbf{X} = [X, Y, Z : 1]^T$,

$$\mathbf{A} = \begin{bmatrix} X_1 & X_2 & X_3 & X_4 \\ Y_1 & Y_2 & Y_3 & Y_4 \\ Z_1 & Z_2 & Z_3 & Z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \text{ and } \mathbf{B} = [\mathbf{A} \mid \mathbf{X}]$$

Again, we are looking for a vector $\boldsymbol{\tau}$ that satisfies the equation

$$\boldsymbol{\tau}^T \mathbf{b} = 0$$

where: $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3, \tau_4 : \tau_5]^T$

The equation can be expressed using a determinant form as:

$$\det \begin{vmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 & \tau_5 \\ X_1 & X_2 & X_3 & X_4 & X \\ Y_1 & Y_2 & Y_3 & Y_4 & Y \\ Z_1 & Z_2 & Z_3 & Z_4 & Z \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix} = 0$$

It can be seen that we can formally write again:

$$\mathbf{b} = \boldsymbol{\xi} \times \boldsymbol{\eta} \times \boldsymbol{\zeta} \times \mathbf{w}$$

where: $\mathbf{b} = [b_1, b_2, b_3, b_4 : b_5]^T$ $\boldsymbol{\xi} = [X_1, X_2, X_3, X_4 : X]^T$

$\boldsymbol{\eta} = [Y_1, Y_2, Y_3, Y_4 : Y]^T$ $\boldsymbol{\zeta} = [Z_1, Z_2, Z_3, Z_4 : Z]^T$

$\mathbf{w} = [1, 1, 1, 1 : 1]^T$

This means that the barycentric coordinates of the point $\mathbf{X}$ are given as:

$$a_1 = -\frac{b_1}{b_5}, \quad a_2 = -\frac{b_2}{b_5},$$

$$a_3 = -\frac{b_3}{b_5}, \quad a_4 = -\frac{b_4}{b_5}$$

or if we use the Plücker coordinates notation, they are given as

$a_i = (-b_i : b_5)$ $i = 1, \ldots, 4$.

The given point $\mathbf{X}$ is inside the given tetrahedron **if and only if** $0 \le a_i \le 1$, i = 1,…,4.

This condition can be expressed by the following sequence

if $b_5 > 0$ **then** $0 \le -b_i \le b_5$

else $b_5 \le -b_i \le 0$

If $b_5 = 0$, the tetrahedron is degenerated to a triangle or to a line segment or to a point, i.e. singular cases that can be correctly detected.

Let us again consider a case when the tetrahedron vertices and the given point are in homogeneous coordinates.

The linear system of equations can be rewritten as:

$$a_1 \frac{x_1}{w_1} + a_2 \frac{x_2}{w_2} + a_3 \frac{x_3}{w_3} + a_4 \frac{x_4}{w_4} = \frac{x}{w}$$

$$a_1 \frac{y_1}{w_1} + a_2 \frac{y_2}{w_2} + a_3 \frac{y_3}{w_3} + a_4 \frac{y_4}{w_4} = \frac{y}{w}$$

$$a_1 \frac{z_1}{w_1} + a_2 \frac{z_2}{w_2} + a_3 \frac{z_3}{w_3} + a_4 \frac{z_4}{w_4} = \frac{z}{w}$$

$$a_1 + a_2 + a_3 + a_4 = 1$$

where: $\mathbf{x}_i = [x_i, y_i, z_i : w_i]^T$ represents the i-th vertex coordinates in the homogeneous coordinates.

We can multiply the linear system of equations by $w \neq 0$, $w_i \neq 0$ $i = 1, \ldots, 4$ and substitute

$$b_1 = -a_1 w_2 w_3 w_4 w \qquad b_2 = -a_2 w_1 w_3 w_4 w$$

$$b_3 = -a_3 w_1 w_2 w_4 w \qquad b_4 = -a_4 w_1 w_2 w_3 w$$

$$b_5 = w_1 w_2 w_3 w_4$$

This results into a standard homogeneous linear system:

$$b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4 + b_5 x = 0$$

$$b_1 y_1 + b_2 y_2 + b_3 y_3 + b_4 y_4 + b_5 y = 0$$

$$b_1 z_1 + b_2 z_2 + b_3 z_3 + b_4 z_4 + b_5 z = 0$$

$$w_1 b_1 + w_2 b_2 + w_3 b_3 + w_4 b_4 + w\, b_5 = 0$$

that can be expressed in the matrix form as:

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x \\ y_1 & y_2 & y_3 & y_4 & y \\ z_1 & z_2 & z_3 & z_4 & z \\ w_1 & w_2 & w_3 & w_4 & w \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \mathbf{0}$$

We are looking for a vector $\boldsymbol{\tau}$ that satisfies the equation

$$\boldsymbol{\tau}^T \mathbf{b} = 0$$

where: the vector $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3, \tau_4 : \tau_5]^T$ is defined as

$$\det \begin{vmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 & \tau_5 \\ x_1 & x_2 & x_3 & x_4 & x \\ y_1 & y_2 & y_3 & y_4 & y \\ z_1 & z_2 & z_3 & z_4 & z \\ w_1 & w_2 & w_3 & w_4 & w \end{vmatrix} = 0$$

It can be seen that we can formally write:

$$\mathbf{b} = \boldsymbol{\xi} \times \boldsymbol{\eta} \times \boldsymbol{\zeta} \times \mathbf{w}$$

where: $\mathbf{b} = [b_1, b_2, b_3, b_4 : b_5]^T$ $\boldsymbol{\xi} = [x_1, x_2, x_3, x_4 : x]^T$

$$\boldsymbol{\eta} = \left[y_1, y_2, y_3, y_4 : y \right]^T \quad \boldsymbol{\zeta} = \left[z_1, z_2, z_3, z_4 : z \right]^T$$
$$\mathbf{w} = \left[w_1, w_2, w_3, w_4 : w \right]^T$$

The conditions – if the point is inside the given triangle – are slightly more complex and the condition $0 \le a_i \le 1 \;\; i = 1,\ldots,4$ can be expressed by the following criteria:

$$0 \le \left(-b_1 : w_2 w_3 w_4 w \right) \le 1$$
$$0 \le \left(-b_2 : w_1 w_3 w_4 w \right) \le 1$$
$$0 \le \left(-b_3 : w_1 w_2 w_4 w \right) \le 1$$
$$0 \le \left(-b_4 : w_1 w_2 w_3 w \right) \le 1$$

It is worth noting that the equations for the computation of barycentric coordinates given above can be simplified for special cases, e.g. if the tetrahedron vertices are expressed in the Euclidean coordinates or the given point **x** is expressed in the Euclidean coordinates. Such simplifications will increase the speed of computation significantly without compromising the robustness of the computation. Nevertheless, the resulting barycentric coordinates are generally in the projective space, i.e. the homogeneous coordinate is not equal to '1' in general.

5.2 Dual Barycentric Coordinates

Barycentric coordinates of a point $\boldsymbol{x} = [x, y: w]^T$ in the triangle given by points $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}$ in E^2 can be computed directly using homogeneous coordinates as $\tilde{\boldsymbol{x}} \times \tilde{\boldsymbol{y}} \times \tilde{\boldsymbol{w}}$,
where: $\tilde{\boldsymbol{x}} = [x_1, x_2, x_3, x]^T$, $\tilde{\boldsymbol{y}} = [y_1, y_2, y_3, y]^T$, $\tilde{\boldsymbol{x}} = [w_1, w_2, w_3, w]^T$

$$\tilde{\boldsymbol{x}} \times \tilde{\boldsymbol{y}} \times \tilde{\boldsymbol{w}} = det \begin{pmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} \\ x_1 & x_2 & x_3 & x \\ y_1 & y_2 & y_3 & y \\ w_1 & w_2 & w_3 & w \end{pmatrix} = [\xi_1 \quad \xi_2 \quad \xi_3 \quad \xi_w]^T$$

where: $\lambda_i = -\xi_i / \xi_w, i = 1, \ldots, 3$ [24].

The P area of a triangle given by three points in E^2 can be easily computed as

$$P = \frac{1}{2} \boldsymbol{x_1}^T . (\boldsymbol{x_2} \times \boldsymbol{x_3}) / (w_1 w_2 w_3) = det \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ w_1 & w_2 & w_3 \end{pmatrix} \Big/ (w_1 w_2 w_3)$$

We can rewrite the result in the projective notation as "projective" scalar value as:

$$P = [det \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ w_1 & w_2 & w_3 \end{pmatrix} : w_1 w_2 w_3]^T$$

Similarly a volume of a tetrahedron given by four points in E^3 can be computed as

$$V = \frac{1}{6} \boldsymbol{x_1}^T . (\boldsymbol{x_2} \times \boldsymbol{x_3} \times \boldsymbol{x_4}) / (w_1 w_2 w_3 w_4)$$

It means that the projective formulation is simple and matrix-vector GPU architecture supports fast computations without using division operation, as the result can be represented by homogeneous coordinates, in general.

As the principle of duality is valid, one could ask: *What is a "dual" value G to a computation of the area P if the triangle is given by three lines in the "normalized" form, e.g.* $\boldsymbol{a_1}^T . (\boldsymbol{a_2} \times \boldsymbol{a_3})$ *instead of three points?*

$$G = \boldsymbol{a_1^T} . (\boldsymbol{a_2} \times \boldsymbol{a_3}) = det \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix} = det \begin{pmatrix} cos\alpha_1 & cos\alpha_2 & cos\alpha_3 \\ sin\alpha_1 & sin\alpha_2 & sin\alpha_3 \\ d_1 & d_2 & d_3 \end{pmatrix}$$

It can be seen that we can apply some transformations so that one vertex of the given triangle is in the origin and the line $\boldsymbol{a_1}$ is on the axis x, the edge $\boldsymbol{a_2}$ passes the origin and line $\boldsymbol{a_3}$ is in the general position.

$$G = (\boldsymbol{T a_1})^T (\boldsymbol{T}^{-1})^T (\boldsymbol{a_2} \times \boldsymbol{a_3}) / det(\boldsymbol{T}) = \boldsymbol{a_1}^T \boldsymbol{T}^T (\boldsymbol{T}^{-1})^T (\boldsymbol{a_2} \times \boldsymbol{a_3}) / det(\boldsymbol{T}) = \boldsymbol{a_1}^T (\boldsymbol{a_2} \times \boldsymbol{a_3}) / det(\boldsymbol{T})$$

As for the "standard" transformations $\det(\boldsymbol{T}) = 1$ and we can write:

$$G = det \begin{pmatrix} 1 & cos\alpha_2 & cos\alpha_3 \\ 0 & sin\alpha_2 & sin\alpha_3 \\ 0 & 0 & d_3 \end{pmatrix} = d_3 sin\alpha_2 = d_3 . a/(2R) = P/R$$

It can be seen that $G = d_3 sin\alpha_2 = P/R$, where: a is the length of the line segment on $\boldsymbol{a_3}$ and R is a radius of the circumscribing circle. It can be seen that the value G can be used as criterion for a quality triangular meshes.

Of course, we have to prove that the proposed transformation of the given triangle is invariant to the G value. As $det(\boldsymbol{T}) = 1$ for translation and rotation operations, those transformations are invariant and value G is not changed by those transformations. The value G has a property of a distance, i.e. it is in measured in [m], in general.

In geometric modeling a skewnees factor S is used for quality evaluation of triangular meshes

$$S = 1 - 2r/R$$

where r is a radius of the inscribed circle.

It seems to that the value G can be used for an effective evaluation for quality of triangular meshes in E^2 or tetrahedron meshes in E^3.

6 Conclusion

This paper briefly described some problems in numerical computations, advantages of the projective space representation use and some well known disasters caused by impropriate use on numerical computations. Geometrical applications and computational methods require robust algorithms. Some above described principles have been applied recently, e.g. in clipping algorithms [23], [34], [39].

The projective space representation and reformulation of geometrical problems leads to more robust algorithms and simpler formulations as shown. The matrix-vector operations lead to more compact algorithms and due to the today's hardware also to computation acceleration, especially if GPU is used.

7 Acknowledgments

The author would like to express his thanks to MSc. students and colleagues at the University of West Bohemia, Plzen and to Dr.Rongjinang Pan, Shandong University, China for their critical and constructive comments and suggestions for their suggestions that helped to improve the manuscript.

This work was supported by the Ministry of Education – projects No.ME10060 and LH12181.

References:

[1] Cuyt,A., Verdonk,B., Becuwe,S., Kuterna,P.: A remarkable Example of Catastrophics Cancellation Unraveled, Computing 66, pp.309-320, 2011

[2] Fu,H., Mencer,O., Luk,W.: Comparing Floating-point and Logarithmic Number Representations for Reconfigurable Acceleration

[3] Gibson,C.G., Hunt,K.H.: Geometry of Screw Systems, Mechanical Machine Theory, Vol.12, pp.1-27, 1992

[4] Hanrahan,P.: Ray-Triangle and Ray-Quadrirateral Intersections in Homogeneous Coordinates, http://graphics.stanford.edu/courses/cs348b-04/rayhomo.pdf, (unpublished) 1989

[5] Hartley,R, Zisserman,A.: MultiView Geometry in Computer Vision, Cambridge Univ. Press, 2000.

[6] Hildenbrand,D., Fontijne,D., Perwass,C., Dorst,L: Geometric Algebra and its Application to Computer Graphics, Eurographics 2004 Tutorial, pp.1-49, 2004.

[7] Hill,F.S.: Computer Graphics using OpenGL, Prentice Hall, pp.827, 2001

[8] Hradek,J., Skala,V.: Hash Function and Triangular Mesh Reconstruction, Vol.29, Computers&Geosciences, Pergamon Press, No.6., pp.741-751, 2003

[9] Chevalley,C.: Fundamental Concepts of Algebra, Academic Press, pp.201-203, 1956,

[10] Jimenez,J.J., Segura,R.J., Feito,F.R.: Efficient Collision Detection between 2D Polygons, Journal of WSCG, Vol.12, No.1-3, 2003

[11] Johnson,M.: Proof by Duality: or the Discovery of "New" Theorems, Mathematics Today, December 1996.

[12] Klein,F.: Notiz Betreffend dem Zusammenhang der Liniengeometrie mit der Mechanik starrer KÄorper.,Math. Ann. 4 , pp.403- 415, 1871

[13] Leclerc,A.P.: Efficient and Reliable Global Optimization, PhD Thesis, Ohio State University, 1992

[14] Lorentzen,L.: Continued Fractions, Atlanties Studies in Mathematics for Engineering and Science, World Scientific Publ., 2008

[15] Ma,Y., Soatto,S., Kosecka,J., Sastry,S.S.: An Invitation to 3D Vision, Springer Verlag, 2004

[16] Matousek,R., Tichy,M., Pohl,Z., Kadlec,J., Softley,C., Coleman,N.: Logarithmic Number System and Floating-point Arithmetic on FPGA, in Proc. FPL, 2002, pp. 627–636.

[17] Mohr,R., Triggs,B.: Projective Geometry for Image Analysis, Tutorial notes, http://www.inrialpes.fr/movi, 1996

[18] Oh,E., Walster,W.G.: Rump's Example Revisited, Reliable Computing, Kluwer Academic Publ., Vol.9., pp.245-248, 2002

[19] Press,W.H., Teukolsky,S.A., Vetterling,W.T., Flannery,B.P.: Numerical recipes in C, Cambridge University Press, 1999

[20] Rump,S.M.: Realiability in Computing, The role of Interval Methods in Scientific Computing, Academic Press, 1988

[21] Skala,V., Kaiser,J., Ondracka,V.: Library for Computation in the Projective Space, 6th Int.Conf. Aplimat, Bratislava, pp. 125-130, 2007

[22] Skala,V., Ondracka,V.: A Precision of Computation in the Projective Space, Recent

Researches in Computer Science, pp.35-40, 15th WSEAS Int.Conference on Computers, Corfu, Greece, 2011
[23] Skala,V.: A New Line Clipping Algorithm with Hardware Acceleration, CGI'2004 conference proceedings, IEEE, Greece, 2004
[24] Skala,V.: Barycentric Coordinates Computation in Homogeneous Coordinates, Computers & Graphics, Elsevier, Vol. 32, No.1, pp.120-127, 2008
[25] Skala,V.: Computation in Projective Space, MAMECTIS conference, La Laguna, Spain, WSEAS, pp.152-157, 2009
[26] Skala,V.: Duality and Intersection Computation in Projective Space with GPU support, Applied Mathematics, Simulation and Modeling - ASM 2010 conference, NAUN, Corfu, Greece, pp.66-71, 2010
[27] Skala,V.: Geometric Computation, Duality and Projective Space, ICGG 2010 conference, pp.363-364, Kyoto, Japan, 2010
[28] Skala,V.: Geometric Computation, Duality and Projective Space, IW-LGK workshop proceedings, pp.105-111, Dresden University of Technology, 2011
[29] Skala,V.: Intersection Computation in Projective Space using Homogeneous Coordinates, Int.Journal on Image and Graphics, Vol.8, No.4, pp.615-628, 2008
[30] Skala,V.: Length, Area and Volume Computation in Homogeneous Coordinates, International Journal of Image and Graphics, Vol.6., No.4, pp.625-639, 2006.
[31] Skala,V.: Projective Geometry and Duality for Graphics, Games and Visualization, - Course notes SIGGRAPH Asia, accepted for publication, 2012.
[32] Skala,V.: Data Interpolation and Transformation of Parametric Patches, WSEAS Visualization, Imaging and Simulation VIS'12, Malta, pp.175-181, 2012.
[33] Skala,V.: Mathematical Foundations for Computer Graphics and Vision and Computations in Projective Spaces, Tutorial 3DTV conference, 2007.
[34] Skala,V: A new Approach to Line and Line Segment Clipping in Homogeneous Coordinates, The Visual Computer, Vol.21, No.11, pp.905-914, 2005
[35] Skala,V: Duality and Intersection Computation in Projective Space with GPU Support, WSEAS Trans.on Mathematics, Vol.9., No.6., pp.407-416, 2010
[36] Skala,V.: A Unified Approach for Textual and Geometrical Information Retrieval, 16th WSEAS Int.Conf. on Computers ICCOMP12, Kos, Greece, pp.143-148, 2012
[37] Skala,V., Hradek,J., Kuchar,M.: New Hash Fuction Construction for Textual and Geometrical Data Retrieval, Latest Trends on Computers, CSCC conference, Corfu, Vol.2, pp.483-489, Greece, 2010
[38] Stolfi,J.: Oriented Projective Geometry, Academic Press, 2001.
[39] Thomas,F., Torras,C.: A Projective invariant intersection test for polyhedra, The Visual Computer, Vol.18, No.1, pp.405-414, 2002
[40] Vince,J.: Geometry for Computer Graphics, Springer Verlag, 2004
[41] Yamaguchi,F., Niizeki,M.: Some basic geometric test conditions in terms of Plücker coordinates and Plücker coefficients, The Visual Computer, Vol.13, pp.29-41, 1997
[42] Yamaguchi,F.: Computer-Aided geometric Design – A Totally Four Dimensional Approach, Springer Verlag, 1996
[43] Zapletal,J., Vanecek,P., Skala,V.: RBF-based Image Restoration Utilising Auxiliary Points, CGI 2009 proceedings, pp.39-44, 2009

WEB resources

[44] Ackermann function, http://en.wikipedia.org/wiki/Ackermann_function, <retrieved 2012-02-23>
[45] Ariane 501 report http://www.esa.int/esaCP/Pr_33_1996_p_EN.html <retrieved 2012-02-02>
[46] Arnold,D.A.: The sinking of the Sleipner A offshore Platform http://www.ima.umn.edu/~arnold/disasters/sleipner.html <retrieved 2012-02-02>
[47] Arnold,D.A.: Two disasters caused by computer arithemtic error, http://www.ima.umn.edu/~arnold/455.f96/disasters.html <retrieved 2012-02-02>
[48] Continuous Fractions http://www.numericana.com/answer/fractions.htm <retrieved 2012-01-29>
[49] IEEE-754 Data Format, http://en.wikipedia.org/wiki/IEEE_754-2008 <retrieved 2012-01-29>
[50] Patriot Missile Defense: Software Problem Led to System Failure at Hhara, Saudi Arabia, GAO/IMTEC-92-26 Report, House of Representatives , USA, 2009

http://www.gao.gov/assets/220/215614.pdf <retrieved 2012-01-29>
[51] SINTEF, http://www.sintef.no/ <retrieved 2012-02-23>
[52] Tucker,W.: Automatic Differentiation, http://www.sintef.no/project/eVITAmeeting/2010/vn2010.pdf <retrieved 2012-01-29>
[53] Materials Digital Library Pathway - MATDL, http://matdl.org/failurecases/Main_Page, <retrieved 2012-02-23>

Appendix A

The cross product in 4D is defined as

$$\mathbf{x}_1 \times \mathbf{x}_2 \times \mathbf{x}_3 = \det \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{vmatrix}$$

and can be implemented in Cg/HLSL on a GPU as follows:

```
float4 cross_4D(float4 x1, float4 x2, float4 x3)
{      float4 a;
a.x = dot(x1.yzw, cross(x2.yzw, x3.yzw));
a.y = - dot(x1.xzw, cross(x2.xzw, x3.xzw));
a.z = dot(x1.xyw, cross(x2.xyw, x3.xyw));
a.w = - dot(x1.xyz, cross(x2.xyz, x3.xyz));
return a;
}
```

or more compactly

```
float4 cross_4D(float4 x1, float4 x2, float4 x3)
{
return (
  dot(x1.yzw, cross(x2.yzw, x3.yzw)),
- dot(x1.xzw, cross(x2.xzw, x3.xzw)),
  dot(x1.xyw, cross(x2.xyw, x3.xyw)),
- dot(x1.xyz, cross(x2.xyz, x3.xyz)) );
}
```