"Extended Cross-Product" and Solution of a Linear System of Equations

Vaclav Skala(✉)

Faculty of Applied Sciences, University of West Bohemia,
Univerzitni 8, 30614 Plzen, Czech Republic
skala@kiv.zcu.cz
http://www.VaclavSkala.euss

Abstract. Many problems, not only in computer vision and visualization, lead to a system of linear equations $\boldsymbol{Ax} = \boldsymbol{0}$ or $\boldsymbol{Ax} = \boldsymbol{b}$ and fast and robust solution is required. A vast majority of computational problems in computer vision, visualization and computer graphics are three dimensional in principle. This paper presents equivalence of the cross–product operation and solution of a system of linear equations $\boldsymbol{Ax} = \boldsymbol{0}$ or $\boldsymbol{Ax} = \boldsymbol{b}$ using projective space representation and homogeneous coordinates. This leads to a conclusion that division operation for a solution of a system of linear equations is not required, if projective representation and homogeneous coordinates are used. An efficient solution on CPU and GPU based architectures is presented with an application to barycentric coordinates computation as well.

Keywords: Linear system of equations · Extended cross-product · Projective space computation · Geometric algebra · Scientific computation

1 Introduction

Many applications, not only in computer vision, require a solution of a homogeneous system of linear equations $\boldsymbol{Ax} = \boldsymbol{0}$ or a non-homogeneous system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$. There are several numerical methods used implemented in standard numerical libraries. However, the numerical solution actually does not allow further symbolic manipulation. Even more, solutions of equations $\boldsymbol{Ax} = \boldsymbol{0}$ and $\boldsymbol{Ax} = \boldsymbol{b}$ are considered as different problems and especially $\boldsymbol{Ax} = \boldsymbol{0}$ is not usually solved quite correctly as users tend to use some additional condition for $\boldsymbol{x}$ unknown (usually setting $x_k = 1$ or so).

In the following, we show the equivalence of the extended cross-product (outer product or progressive product) with a solution of both types of linear systems of equations, i.e. $\boldsymbol{Ax} = \boldsymbol{0}$ and $\boldsymbol{Ax} = \boldsymbol{b}$.

Many problems in computer vision, computer graphics and visualization are 3-dimensional. Therefore specific numerical approaches can be applied to speed up the solution. In the following extended cross-product, also called outer product or progressive product, is introduced in the "classical" notation using "×" symbol.

O. Gervasi et al. (Eds.): ICCSA 2016, Part I, LNCS 9786, pp. 18–35, 2016.
DOI: 10.1007/978-3-319-42085-1_2

2 Extended Cross Product

Let us consider the standard cross-product of two vectors $\boldsymbol{a} = [a_1, a_2, a_3]^T$ and $\boldsymbol{b} = [b_1, b_2, b_3]^T$. Then the cross-product is defined as:

$$\boldsymbol{a} \times \boldsymbol{b} = \det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} \tag{1}$$

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

If a matrix form is needed, then we can write:

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2}$$

In some applications the matrix form is more convenient.

Let us introduce the extended cross-product of three vectors $\boldsymbol{a} = [a_1, \ldots, a_n]^T$, $\boldsymbol{b} = [b_1, \ldots, b_n]^T$ and $\boldsymbol{c} = [c_1, \ldots, c_n]^T$, $n = 4$ as:

$$\boldsymbol{a} \times \boldsymbol{b} \times \boldsymbol{c} = \det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} \\ a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{bmatrix} \tag{3}$$

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

It can be shown that there exists a matrix form for the extended cross-product representation:

$$\boldsymbol{a} \times \boldsymbol{b} \times \boldsymbol{c} = (-1)^{n+1} \begin{bmatrix} 0 & -\delta_{34} & \delta_{24} & -\delta_{23} \\ \delta_{34} & 0 & -\delta_{14} & \delta_{13} \\ -\delta_{24} & \delta_{14} & 0 & -\delta_{12} \\ \delta_{23} & -\delta_{13} & \delta_{12} & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \tag{4}$$

where: $n = 4$. In this case and δ_{ij} are sub-determinants with columns i, j of the matrix $\boldsymbol{T}$ defined as:

$$\boldsymbol{T} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix} \tag{5}$$

e.g. sub-determinant $\delta_{24} = \det \begin{bmatrix} a_2 & a_4 \\ b_2 & b_4 \end{bmatrix}$ etc.

The extended cross-product for 5-dimensions is defined as:

$$\boldsymbol{a} \times \boldsymbol{b} \times \boldsymbol{c} \times \boldsymbol{d} = \det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} & \boldsymbol{n} \\ a_1 & a_2 & a_3 & a_4 & a_5 \\ b_1 & b_2 & b_3 & b_4 & b_5 \\ c_1 & c_2 & c_3 & c_4 & c_5 \\ d_1 & d_2 & d_3 & d_4 & d_5 \end{bmatrix} \tag{6}$$

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

It can be shown that there exists a matrix form as well:

$$\boldsymbol{a} \times \boldsymbol{b} \times \boldsymbol{c} \times \boldsymbol{d} = (-1)^{n+1} \begin{bmatrix} 0 & -\delta_{345} & \delta_{245} & -\delta_{235} & \delta_{234} \\ \delta_{345} & 0 & -\delta_{145} & \delta_{135} & -\delta_{134} \\ -\delta_{245} & \delta_{145} & 0 & -\delta_{125} & \delta_{124} \\ \delta_{235} & -\delta_{135} & \delta_{125} & 0 & -\delta_{123} \\ -\delta_{234} & \delta_{134} & -\delta_{124} & \delta_{123} & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} \tag{7}$$

where $n = 5$. In this case and δ_{ijk} are sub-determinants with columns *i, j, k* of the matrix $\boldsymbol{T}$ defined as:

$$\boldsymbol{T} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ b_1 & b_2 & b_3 & b_4 & b_5 \\ c_1 & c_2 & c_3 & c_4 & c_5 \end{bmatrix} \tag{8}$$

e.g. sub-determinant δ_{245} is defined as:

$$\delta_{245} = \det \begin{bmatrix} a_2 & a_4 & a_5 \\ b_2 & b_4 & b_5 \\ c_2 & c_4 & c_5 \end{bmatrix} = a_2 \det \begin{bmatrix} b_4 & b_5 \\ c_4 & c_5 \end{bmatrix} - a_4 \det \begin{bmatrix} b_2 & b_5 \\ c_2 & c_5 \end{bmatrix} + a_5 \det \begin{bmatrix} b_2 & b_4 \\ c_2 & c_4 \end{bmatrix} \tag{9}$$

In spite of the "complicated" description above, this approach leads to a faster computation in the case of lower dimensions, see Sect. 7.

3 Projective Representation and Duality Principle

Projective representation and its application for computation are considered to be mysterious or too complex. Nevertheless we are using it naturally very frequently in the form of fractions, e.g. *a/b*. We also know that fractions help us to express values, which cannot be expressed precisely due to limited length of a mantissa, e.g. $1/3 = 0{,}33\ldots\ldots333\ldots = 0.\bar{3}$.

In the following we will explore projective representation, actually rational fractions, and its applicability.

3.1 Projective Representation

Projective extension of the Euclidean space is used commonly in computer graphics and computer vision mostly for geometric transformations. However, in computational sciences, the projective representation is not used, in general. This chapter shortly introduces basic properties and mutual conversions. More detailed description of projective representation and applications can be found in [12, 15, 20].

The given point $\boldsymbol{X} = (X,Y)$ in the Euclidean space E^2 is represented in homogeneous coordinates as $\boldsymbol{x} = [x, y : w]^T$, $w \neq 0$. It can be seen that $\boldsymbol{x}$ is actually a line in the projective space P^3 with the origin excluded. Mutual conversions are defined as:

$$X = \frac{x}{w} \qquad Y = \frac{y}{w} \tag{10}$$

where: $w \neq 0$ is the homogeneous coordinate. Note that the homogeneous coordinate w is actually a scaling factor with no physical meaning, while x, y are values with physical units in general.

The projective representation enables us nearly double precision as the mantissa of x, resp. y and w are used for a value representation. However we have to distinguish two different data types, i.e.

- Projective representation of a $\boldsymbol{n}$-dimensional value $\boldsymbol{X} = (X_1, \ldots, X_n)$, represented by one dimensional array $\boldsymbol{x} = [x_1, \ldots, x_n : x_w]^T$, e.g. coordinates of a point, that is fixed to the origin of the coordinate system.
- Projective representation of a n -dimensional vector (in the mathematical meaning) $\boldsymbol{A} = (A_1, \ldots, A_n)$, represented by one dimensional array $\boldsymbol{a} = [a_1, \ldots, a_n : a_w]^T$. In this case the homogeneous coordinate a_w is actually just a scaling factor. Any vector is not fixed to the origin of the coordinate system and it is "movable".

Therefore a user should take an attention to the correctness of operations. Another interesting application of the projective representation is the rational trigonometry [19].

3.2 Principle of Duality

The projective representation offers also one very important property – principle of duality. The principle of duality in E^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 [1, 5, 9, 14]. 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. it can be shown that operations "join" a "meet" are dual as well.

This helps a lot to solve some geometrical problems. In the following we will demonstrate that on very simple geometrical problems like intersection of two lines, resp. three planes and computation of a line given by two points, resp. of a plane given by three points.

4 Solution of $Ax = B$

Solution of non-homogeneous system of equation $\boldsymbol{AX} = \boldsymbol{b}$ is used in many computational tasks.

For simplicity of explanation, let us consider a simple example of intersection computation of two lines p_1 a p_2 in E^2 given as:

$$p_1 : A_1X + B_1Y + C_1 = 0 \qquad p_2 : A_2X + B_2Y + C_2 = 0 \tag{11}$$

An intersection point of two those lines is given as a solution of a linear system of equations: $\boldsymbol{Ax} = \boldsymbol{b}$:

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} -c_1 \\ -c_2 \end{bmatrix} \tag{12}$$

Generally, for the given system of $\boldsymbol{n}$ liner equations with $\boldsymbol{n}$ unknowns in the form $\boldsymbol{AX} = \boldsymbol{b}$ the solution is given:

$$X_i = \frac{\det(\boldsymbol{A}_i)}{\det(\boldsymbol{A})} \quad i = 1, \ldots, n \tag{13}$$

where: $\boldsymbol{A}$ is a regular matrix $n \times n$ having non-zero determinant, the matrix $\boldsymbol{A}_i$ is the matrix $\boldsymbol{A}$ with replaced i^{th} column by the vector $\boldsymbol{b}$ and $\boldsymbol{X} = [X_1, \ldots, X_n]^T$ is a vector of unknown values.

In a low dimensional case using general methods for solution of linear equations, e.g. Gauss-Seidel elimination etc., is computational expensive. Also division operation is computationally expensive and decreasing precision of a solution.

Usually, a condition **if** $\det(\boldsymbol{A}) < eps$ **then** EXIT is taken for solving "close to singular cases". Of course, nobody knows, what a value of *eps* is appropriate.

5 Solution of $Ax = 0$

There is another very simple geometrical problem; determination of a line p given by two points $\boldsymbol{X}_1 = (X_1, Y_1)$ and $\boldsymbol{X}_2 = (X_2, Y_2)$ in E^2. This seems to be a quite simple problem as we can write:

$$aX_1 + bY_1 + c = 0 \qquad aX_2 + bY_2 + c = 0 \tag{14}$$

i.e. it leads to a solution of homogeneous systems of equations $\boldsymbol{AX} = \boldsymbol{0}$, i.e.:

$$\begin{bmatrix} X_1 & Y_1 & 1 \\ X_2 & Y_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{15}$$

In this case, we obtain one parametric set of solutions as the Eq. (15) can be multiplied by any value $q \neq 0$ and the line is the same.

There is a problem – we know that lines and points are dual in the E^2 case, so the question is why the solutions are not dual. However if the projective representation is used the duality principle will be valid, as follows.

6 Solution $Ax = b$ and $Ax = 0$

Let us consider again intersection of two lines $\boldsymbol{p}_1 = [a_1, b_1 : c_1]^T$ a $\boldsymbol{p}_2 = [a_2, b_2 : c_2]^T$ leading to a solution of non-homogeneous linear system $\boldsymbol{AX = b}$, which is given as:

$$p_1 : a_1X + b_1Y + c_1 = 0 \qquad p_2 : a_2X + b_2Y + c_2 = 0 \tag{16}$$

If the equations are multiplied by $w \neq 0$ we obtain:

$$\begin{aligned} p_1 : a_1X + b_1Y + c_1 &\triangleq & p_2 : a_2X + b_2Y + c_2 &\triangleq \\ a_1x + b_1y + c_1w &= 0 & a_2x + b_2y + c_2w &= 0 \end{aligned} \tag{17}$$

where: $\triangleq$ means "projectively equivalent to" as $x = wX$ and $y = wY$.

Now we can rewrite the equations to the matrix form as $\boldsymbol{Ax = 0}$:

$$\begin{bmatrix} a_1 & b_1 & -c_1 \\ a_2 & b_2 & -b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{18}$$

where $\boldsymbol{x} = [x, y : w]^T$ is the intersection point in the homogeneous coordinates.

In the case of computation of a line given by two points given in homogeneous coordinates, i.e. $\boldsymbol{x}_1 = [x_1, y_1 : w_1]^T$ and $\boldsymbol{x}_2 = [x_2, y_2 : w_2]^T$, the Eq. (14) is multiplied by $w_i \neq 0$.Then, we get a solution in the matrix form as $\boldsymbol{Ax = 0}$, i.e.

$$\begin{bmatrix} x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{19}$$

Now, we can see that the formulation is leading in the both cases to the same numerical problem: to a solution of a homogeneous linear system of equations.

However, a solution of homogeneous linear system of equations is not quite straightforward as there is a one parametric set of solutions and all of them are projectively equivalent. It can be seen that the solution of Eq. (18), i.e. intersection of two lines in E^2, is equivalent to:

$$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2 \tag{20}$$

and due to the principle of duality we can write for a line given by two points:

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \tag{21}$$

In the three dimensional case we can use extended cross-product [12, 15, 16].

A plane $\rho : aX + bY + cY + d = 0$ given by three points $\boldsymbol{x}_1 = [x_1, y_1, z_1 : w_1]^T$, $\boldsymbol{x}_2 = [x_2, y_2, z_2 : w_2]^T$ and $\boldsymbol{x}_2 = [x_3, y_3, z_3 : w_3]^T$ is determined in the projective representation as:

$$\boldsymbol{\rho} = [a, b, c : d]^T = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_2 \tag{22}$$

and the intersection point $\boldsymbol{x}$ of three planes points $\boldsymbol{\rho}_1 = [a_1, b_1, c_1 : d_1]^T$, $\boldsymbol{\rho}_2 = [a_2, b_2, c_2 : d_2]^T$ and $\boldsymbol{\rho}_3 = [a_3, b_3, c_3 : d_3]^T$ is determined in the projective representation as:

$$\boldsymbol{x} = [x, y, z : w]^T = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_2 \tag{23}$$

due to the duality principle.

It can be seen that there is no division operation needed, if the result can be left in the projective representation. The approach presented above has another one great advantage as it allows symbolic manipulation as we have avoided numerical solution and also precision is nearly doubled.

7 Barycentric Coordinates Computation

Barycentric coordinates are often used in many engineering applications, not only in geometry. The barycentric coordinates computation leads to a solution of a system of linear equations. However it was shown, that a solution of a linear system equations is equivalent to the extended cross product [12–14]. Therefore it is possible to compute barycentric coordinates using cross product which is convenient for application of SSE instructions or for GPU oriented computations. Let us demonstrate the proposed approach on a simple example again.

Given a triangle in E^2 defined by points $\boldsymbol{x}_i = [x_i, y_i : 1]^T$, $i = 1, \ldots, 3$, the barycentric coordinates of the point $\boldsymbol{x}_0 = [x_0, y_0 : 1]^T$ can be computed as follows:

$$\begin{aligned} \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 &= x_0 \\ \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 &= y_0 \\ \lambda_1 + \lambda_2 + \lambda_3 &= 1 \end{aligned} \tag{24}$$

For simplicity, we set $w_i = 1, i = 1, \ldots, 3$. It means that we have to solve a system of linear equations $\boldsymbol{Ax} = \boldsymbol{b}$:

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{25}$$

if the points are given in the projective space with homogeneous coordinates $\boldsymbol{x}_i = [x_i, y_i : w_i]^T, i = 1, \ldots, 3$ and $\boldsymbol{x}_0 = [x_0, y_0 : w_0]^T$. It can be easily proved, due to the multilinearity, we need to solve a linear system $\boldsymbol{Ax} = \boldsymbol{b}$:

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ w_0 \end{bmatrix} \tag{26}$$

Let us define new vectors containing a row of the matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ as:

$$\boldsymbol{x} = [x_1, x_2, x_3, x_0]^T \quad \boldsymbol{y} = [y_1, y_2, y_3, y_0]^T \quad \boldsymbol{w} = [w_1, w_2, w_3, w_0]^T \tag{27}$$

The projective barycentric coordinates $\boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3 : \xi_w]^T$ are given as:

$$\lambda_1 = -\frac{\xi_1}{\xi_w} \quad \lambda_2 = -\frac{\xi_2}{\xi_w} \quad \lambda_3 = -\frac{\xi_3}{\xi_w} \tag{28}$$

i.e.

$$\lambda_i = -\frac{\xi_i}{\xi_w} \quad i = 1, \ldots, 3 \tag{29}$$

Using the extended cross product, the projective barycentric coordinates are given as:

$$\boldsymbol{\xi} = \boldsymbol{x} \times \boldsymbol{y} \times \boldsymbol{w} = \det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} \\ x_1 & x_2 & x_3 & x_0 \\ y_1 & y_2 & y_3 & y_0 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} = [\xi_1, \xi_2, \xi_3 : \xi_w]^T \tag{30}$$

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

Similarly in the E^3 case, given a tetrahedron in E^3 defined by points $\boldsymbol{x}_i = [x_i, y_i, z_i : w_i]^T$, $i = 1, \ldots, 3$ and the point $\boldsymbol{x}_0 = [x_0, y_0, z_0 : w_0]^T$:

$$\begin{aligned} \boldsymbol{x} &= [x_1, x_2, x_3, x_4 : x_0]^T \quad \boldsymbol{y} = [y_1, y_2, y_3, y_4 : y_0]^T \\ \boldsymbol{z} &= [z_1, z_2, z_3, z_4 : z_0]^T \quad \boldsymbol{w} = [w_1, w_2, w_3, w_4 : w_0]^T \end{aligned} \tag{31}$$

Then projective barycentric coordinates are given as:

$$\boldsymbol{\xi} = \boldsymbol{x} \times \boldsymbol{y} \times \boldsymbol{z} \times \boldsymbol{w} = [\xi_1, \xi_2, \xi_3, \xi_4 : \xi_w]^T \tag{32}$$

The Euclidean barycentric coordinates are given as:

$$\lambda_1 = -\frac{\xi_1}{\xi_w} \quad \lambda_2 = -\frac{\xi_2}{\xi_w} \quad \lambda_3 = -\frac{\xi_3}{\xi_w} \quad \lambda_4 = -\frac{\xi_4}{\xi_w} \tag{33}$$

i.e.

$$\lambda_i = -\frac{\xi_i}{\xi_w} \quad i = 1, \ldots, 4 \tag{34}$$

How Simple and Elegant Solution! The presented computation of barycentric coordinates is simple and convenient for GPU use or SSE instructions. Even more, as we have assumed from the very beginning, there is no need to convert projective values to the Euclidean notation. As a direct consequence of that is, that we are saving a lot of computational time also increasing robustness of the computation, especially due to division operation elimination. As a result is represented as a rational fraction, the precision is nearly equivalent to double mantissa precision and exponent range.

Let us again present advantages of the projective representation on simple examples.

8 Intersection of Two Planes

Intersection of two planes ρ_1 and ρ_1 in E^3 is seemingly a simple problem, but surprisingly computationally expensive, Fig. 1. Let us consider the "standard" solution in the Euclidean space and a solution using the projective approach.

Given two planes ρ_1 and ρ_2 in E^3:

$$\boldsymbol{\rho}_1 = [a_1, b_1, c_1 : d_1]^T = [\boldsymbol{n}_1^T : d_1]^T \quad \boldsymbol{\rho}_2 = [a_2, b_2, c_2 : d_2]^T = [\boldsymbol{n}_2^T : d_2]^T \tag{35}$$

where: $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are normal vectors of those planes.

Then the directional vector $\boldsymbol{s}$ of a parametric line $\boldsymbol{X}(t) = \boldsymbol{X}_0 + \boldsymbol{s}t$ is given by a cross product:

$$\boldsymbol{s} = \boldsymbol{n}_1 \times \boldsymbol{n}_2 \equiv [a_3, b_3, c_3]^T \tag{36}$$

and point $\boldsymbol{X}_0 \in E^3$ of the line is given as:

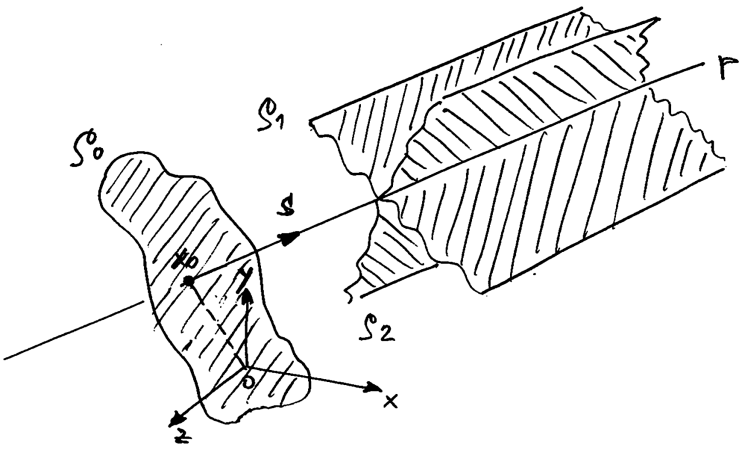

Fig. 1. A line as the intersection of two planes

$$X_0 = \frac{d_2\begin{vmatrix} b_1 & c_1 \\ b_3 & c_3 \end{vmatrix} - d_1\begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix}}{DET} \quad Y_0 = \frac{d_2\begin{vmatrix} a_3 & c_3 \\ a_1 & c_1 \end{vmatrix} - d_1\begin{vmatrix} a_3 & c_3 \\ a_2 & c_2 \end{vmatrix}}{DET}$$
$$Z_0 = \frac{d_2\begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix} - d_1\begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}}{DET} \quad DET = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} \tag{37}$$

It can be seen that the formula above is quite difficult to remember and its derivation is not simple. It should be noted that there is again a severe problem with stability and robustness if a condition like $|DET| < eps$ is used. Also the formula is not convenient for GPU or SSE applications. There is another equivalent solution based on Plücker coordinates and duality application, see [12, 16].

Let us explore a solution based on the projective representation explained above.

Given two planes ρ_1 and ρ_2. Then the directional vector $\boldsymbol{s}$ of their intersection is given as:

$$\boldsymbol{s} = \boldsymbol{n}_1 \times \boldsymbol{n}_2 \tag{38}$$

We want to determine the point $\boldsymbol{x}_0$ of the line given as an intersection of those two planes. Let us consider a plane ρ_0 passing the origin of the coordinate system with the normal vector $\boldsymbol{n}_0$ equivalent to $\boldsymbol{s}$, Fig. 1. This plane ρ_0 is represented as:

$$\boldsymbol{\rho}_0 = [a_0, b_0, c_0 : 0]^T = [\boldsymbol{s}^T : 0]^T \tag{39}$$

Then the point $\boldsymbol{x}_0$ is simply determined as an intersection of three planes ρ_1, ρ_2, ρ_0 as:

$$\boldsymbol{x}_0 = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_0 = [x_0, y_0, z_0 : w_0]^T \tag{40}$$

It can be seen that the proposed algorithm is simple, easy to understand, elegant and convenient for SEE and GPU applications as it uses vector-vector operations.

9 Closest Point on the Line Given as an Intersection of Two Planes

Another example of advantages of the projective notation is finding the closest point on a line given as an intersection of two planes ρ_1 and ρ_2 to the given point $\boldsymbol{\xi} \in E^3$, Fig. 2. The closest point to the given point on an intersection of two planes

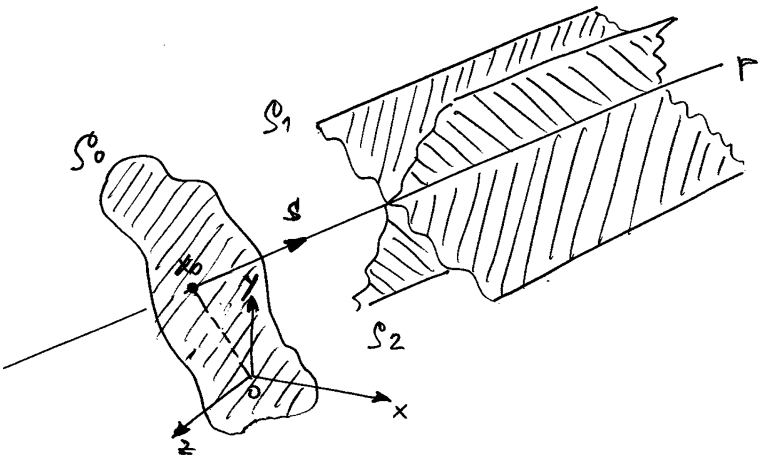

Fig. 2. The closest point to the given point on an intersection of two planes

A solution in the Euclidean space, proposed in [8], is based on a solution of a system of linear equations using Lagrange multipliers, leading to a matrix of (5×5):

$$\begin{bmatrix} 2 & 0 & 0 & n_{1x} & n_{2x} \\ 0 & 2 & 0 & n_{1y} & n_{2y} \\ 0 & 0 & 2 & n_{1z} & n_{2z} \\ n_{1x} & n_{1y} & n_{1z} & 0 & 0 \\ n_{2x} & n_{2y} & n_{2z} & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} 2\xi_x \\ 2\xi_y \\ 2\xi_z \\ \boldsymbol{p}_1\boldsymbol{n}_1 \\ \boldsymbol{p}_2\boldsymbol{n}_2 \end{bmatrix} \tag{41}$$

where: $\boldsymbol{p}_1$, resp. $\boldsymbol{p}_2$ are points on planes ρ_1, resp. ρ_2, with a normal vector $\boldsymbol{n}_1$, resp. $\boldsymbol{n}_2$. Coordinates of the closest point $\boldsymbol{x} = [x, y, z]^T$ on the intersection of two planes to the

point $\boldsymbol{\xi} = (\xi_x, \xi_y, \xi_z)$ are given as a solution of this system of linear equations. Note that the point $\boldsymbol{\xi}$ is given in the Euclidean space.

Let us consider a solution based on the projective representation. The proposed approach is based on basic geometric transformations with the following steps:

1. Translation of planes $\boldsymbol{\rho}_1$, $\boldsymbol{\rho}_2$ and point $\boldsymbol{\xi} = [\xi_x, \xi_y, \xi_z : 1]^T$ so that the point $\boldsymbol{\xi}$ is in the origin of the coordinate system, i.e. using transformation matrix $\boldsymbol{T}$ for the point translation and matrix $(\boldsymbol{T}^T)^{-1} = \boldsymbol{T}^{-T}$ for translation of planes [11, 14, 16].
2. Intersection computation of those two translated planes; the result is a line with the directional vector $\boldsymbol{s}$ and point $\boldsymbol{x}_0$
3. Translation of the point $\boldsymbol{x}_0$ by inverse translation using the matrix $\boldsymbol{T}^{-1}$

The translation matrices are defined as:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & -\xi_x \\ 0 & 1 & 0 & -\xi_y \\ 0 & 0 & 1 & -\xi_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{T}^{-T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \xi_x & \xi_y & \xi_z & 1 \end{bmatrix} \qquad (42)$$

$$\boldsymbol{T}' = \begin{bmatrix} \xi_w & 0 & 0 & -\xi_x \\ 0 & \xi_w & 0 & -\xi_y \\ 0 & 0 & \xi_w & -\xi_z \\ 0 & 0 & 0 & \xi_w \end{bmatrix}$$

If the point $\boldsymbol{\xi}$ is given in the projective space, i.e. $\boldsymbol{\xi} = [\xi_x, \xi_y, \xi_z : \xi_w]^T$, $w \neq 1 \,\&\, w \neq 0$, then the matrix $\boldsymbol{T}$ is given as $\boldsymbol{T}'$.

It can be seen that the computation is more simple, robust and convenient for SSE or GPU oriented applications. It should be noted that the formula is more general as the point $\boldsymbol{\xi}$ can be given in the projective space and no division operations are needed.

10 Symbolic Manipulations

Symbolic manipulations are very important and help to find or simplify computational formulas, avoid singularities etc. As the extended cross-product is an associative and anti-commutative as the cross-product in E^3 similar rules are valid, i.e. in E^3:

$$\begin{aligned} \boldsymbol{a} \times (\boldsymbol{b} + \boldsymbol{c}) &= \boldsymbol{a} \times \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{c} \\ \boldsymbol{a} \times \boldsymbol{b} &= -\boldsymbol{b} \times \boldsymbol{a} \end{aligned} \qquad (43)$$

In the case of the extended cross-product, i.e. in the projective notation P^3 we actually formally have operations in E^4:

$$\begin{aligned} \boldsymbol{a} \times (\boldsymbol{b} + \boldsymbol{c}) \times \boldsymbol{d} &= \boldsymbol{a} \times \boldsymbol{b} \times \boldsymbol{d} + \boldsymbol{a} \times \boldsymbol{c} \times \boldsymbol{d} \\ \boldsymbol{a} \times \boldsymbol{b} \times \boldsymbol{c} &= -\boldsymbol{b} \times \boldsymbol{a} \times \boldsymbol{c} \end{aligned} \tag{44}$$

This can be easily proved by applications of rules for operations with determinants.

However, for general understanding more general theory is to be used – Geometric Algebra [2–4, 6, 7, 10, 18], in which the extended cross-product is called outer product and the above identities are rewritten as:

$$\begin{aligned} \boldsymbol{a} \wedge (\boldsymbol{b} + \boldsymbol{c}) \wedge \boldsymbol{d} &= \boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{d} + \boldsymbol{a} \wedge \boldsymbol{c} \wedge \boldsymbol{d} \\ \boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c} &= -\boldsymbol{b} \wedge \boldsymbol{a} \wedge \boldsymbol{c} \end{aligned} \tag{45}$$

where: "$\wedge$" is an operator of the *outer product*, which is equivalent to the cross-product in E^3. There is also an operator "$\vee$" for the *inner product* which is equivalent to the dot product in E^3.

In geometric algebra *geometric product* is defined as:

$$\boldsymbol{ab} = \boldsymbol{a} \vee \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b} \tag{46}$$

i.e. in the case of E^3 we can write:

$$\boldsymbol{ab} = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{b} \tag{47}$$

and getting some "strange", as a scalar and a vector (actually a bivector) are summed together. But it is a valid result and $\boldsymbol{ab}$ is called *geometric product* [18].

However, if the projective representation is used, we need to be a little bit careful with equivalent operations to the standard operations in the Euclidean space.

11 Example of Application

Let us consider a simple example in 3-dimensional space. Assume, that $\boldsymbol{Ax} = \boldsymbol{b}$ is a system of linear equations, i.e.:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{48}$$

and we want to explore $\xi = \boldsymbol{c} \cdot \boldsymbol{x}$, where $\boldsymbol{c} = [c_1, c_2, c_3]^T$.

In the "standard" approach a system of linear equations has to be solved numerically or symbolic manipulation has to be used. We can rewrite the Eq. (48) using the projective representation as:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & -b_1 \\ a_{21} & a_{22} & a_{23} & -b_2 \\ a_{31} & a_{32} & a_{33} & -b_3 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \\ \bar{x}_w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \; \& \, x_i = \frac{\bar{x}_i}{\bar{x}_w} \tag{49}$$

The conversion to the Euclidean space is given as:

$$x_i = \frac{\bar{x}_i}{\bar{x}_w} \quad i = 1, \ldots, 3 \tag{50}$$

Then using equivalence of the extended cross-product and solution of a linear system of equations we can write:

$$\bar{\boldsymbol{x}} = \bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2 \times \bar{\boldsymbol{a}}_3 \tag{51}$$

where: $\bar{\boldsymbol{x}} = [\bar{x}_1, \bar{x}_2, \bar{x}_3 : \bar{x}_w]^T$, $\bar{\boldsymbol{a}}_i = [a_{i1}, a_{i2}, a_{i3} : -b_i]^T$, $i = 1, \ldots, 3$. It should be noted that the result is actually in the 3-dimensional projective space.

In many cases, the result of computation is not necessarily to be converted to the Euclidean space. If left in the projective representation, we save division operations, increase precision of computation as the mantissa is actually nearly doubled (mantissa of $\bar{x}_i$ and $\bar{x}_w$). Also robustness is increased as well as we haven't made any specific assumptions about collinearity of planes. Let a scalar value $\xi \in E^1$ is given as:

$$\xi = \boldsymbol{c} \cdot \boldsymbol{x} \tag{52}$$

The scalar value ξ can be expressed as a homogeneous vector $\bar{\boldsymbol{\xi}}$ in the projective notation as:

$$\bar{\xi}^T = \left[\bar{\xi} : \bar{\xi}_w\right] \quad \& \quad \bar{\xi}_w = 1 \tag{53}$$

Generally, the value in the Euclidean space is given as $\xi = \frac{\bar{\xi}}{\bar{\xi}_w}$. Extension to the 3-dimensional case is straightforward.

As an example let us consider a test if the given point $\bar{\xi} = \left[\bar{\xi}_1, \bar{\xi}_2, \bar{\xi}_3 : \bar{\xi}_w\right]^T$ lies on a plane given by three points $\boldsymbol{x}_i, i = 1, \ldots, 3$ using projective notation. A plane p is given:

$$\boldsymbol{\rho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3 = [a, b, c : d]^T \tag{54}$$

and the given point has to fulfill condition $\bar{\xi} \cdot \boldsymbol{\rho} = a\bar{\xi}_1 + b\bar{\xi}_2 + c\bar{\xi}_{13} + d\bar{\xi}_w = 0$.

We know that:

$$\boldsymbol{a} \times \boldsymbol{b} \times \boldsymbol{c} = \det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} & \boldsymbol{l} \\ a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{bmatrix} = - \begin{bmatrix} 0 & -\delta_{34} & \delta_{24} & -\delta_{23} \\ \delta_{34} & 0 & -\delta_{14} & \delta_{13} \\ -\delta_{24} & \delta_{14} & 0 & -\delta_{12} \\ \delta_{23} & -\delta_{13} & \delta_{12} & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \tag{55}$$

where: $\boldsymbol{i} = [1,0,0,0]^T, \boldsymbol{j} = [0,1,0,0]^T, \boldsymbol{k} = [0,0,1,0]^T, \boldsymbol{l} = [0,0,0,1]^T$. Then, the test $\boldsymbol{\xi} \cdot \boldsymbol{\rho} = 0$ is actually:

$$[\xi_1, \xi_2, \xi_3 : \xi_w] \begin{bmatrix} 0 & -\delta_{34} & \delta_{24} & -\delta_{23} \\ \delta_{34} & 0 & -\delta_{14} & \delta_{13} \\ -\delta_{24} & \delta_{14} & 0 & -\delta_{12} \\ \delta_{23} & -\delta_{13} & \delta_{12} & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ y_3 \\ z_3 \\ w_3 \end{bmatrix} = 0 \tag{56}$$

It means that we are getting a bilinear form:

$$\bar{\xi}^T \boldsymbol{B} \boldsymbol{x}_3 = 0 \tag{57}$$

where: $\boldsymbol{B}$ is an antisymmetric matrix with a null diagonal. So we can analyze such conditions more deeply in an analytical form. It means that we can explore the formula on a symbolic level. It is also possible to derive some additional information for the ξ value, resp. $\bar{\xi}$ value, if the projective notation is used. This approach can be directly extended do the d-dimensional space using geometry algebra [18].

12 Efficiency of Computation and GPU Code

Let us consider reliability and the cost of computation of the "standard" approach using Cramer's rule using determinants. For the given system of n liner equations with n unknowns in the form $\boldsymbol{Ax} = \boldsymbol{b}$ the solution is given as:

$$X_i = \frac{\det(\boldsymbol{A}_i)}{\det(\boldsymbol{A})} \quad i = 1, \ldots, n \tag{58}$$

In the projective notation using homogeneous coordinates we can actually write $\boldsymbol{x} = [x_1, \ldots, x_n : w]^T$, where: $w = \det(\boldsymbol{A})$ and $x_i = \det(\boldsymbol{A}_i)$, $i = 1, \ldots, n$

The projective representation not only enables to postpone division operations, but also offers some additional advantages as follows. Computing of determinants is quite computationally expensive task. However for 2–4 dimensional cases there are some advantages using the extended cross-product as explained below (Table 1).

Table 1. Cost of determinant computation

Operation	$\mathrm{Det}_{2\times2}$	$\mathrm{Det}_{3\times3}$	$\mathrm{Det}_{4\times4}$	$\mathrm{Det}_{5\times5}$
$\pm$	1	6	24	120
$\times$	2	12	48	240

Table 2. Cost of cross-product computation

Operation	$\boldsymbol{a}\times\boldsymbol{b}$	$\boldsymbol{a}\times\boldsymbol{b}\times\boldsymbol{c}$	$\boldsymbol{a}\times\boldsymbol{b}\times\boldsymbol{c}\times\boldsymbol{d}$
"±"	3	27	159
"×"	6	52	173

Table 3. Cost of cross-product computation with subdeterminants

	$\boldsymbol{a}\times\boldsymbol{b}$	$\boldsymbol{a}\times\boldsymbol{b}\times\boldsymbol{c}$	$\boldsymbol{a}\times\boldsymbol{b}\times\boldsymbol{c}\times\boldsymbol{d}$
$\pm$	3	14	60
$\times$	6	24	77

Generally the computational expenses are given as:

$$\mathrm{Det}_{(k+1)\times(k+1)} = k\,\mathrm{Det}_{k\times k} + k(''\pm'') \tag{59}$$

Total cost of computation if Cramer's rule for generalized is used (Table 2):

Computational expenses for the generalized cross-product matrix based formulation, if partial intermediate computations are used (Table 3).

It means, that for the 2-dimensional and 4-dimensional cases, the expected speed up υ is:

$$\upsilon \cong \frac{Cramer's\ rule}{partial\ summation} \doteq 2 \tag{60}$$

In real implementations on CPU the SSE instructions can be used which are more convenient for vector-vector operations and some steps can be made in parallel. Additional speed up can be achieved by GPU use for computation.

In the case of higher dimension modified standard algorithms can be used including iterative methods [17]. Also as the projective representation nearly doubles precision of computation, if a single precision on GPU is used (only few processors compute in a double precision), the result after conversion to the Euclidean representation is equivalent to the double precision.

13 GPU Code

Many today's computational systems can use GPU support, which allows fast and parallel processing. The above presented approach offers significant speed up as the "standard" cross-product is implemented in hardware as an instruction and the extended cross-product for 4D can be implemented as:

```
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}
```

In general, it can be seen that a solution of linear systems of equations on GPU for a small dimension n is simple, fast and can be performed in parallel.

14 Conclusion

Projective representation is not widely used for general computation as it is mostly considered for as applicable to computer graphics and computer vision field only. In this paper the equivalence of cross-product and solution of linear system of equations has been presented. The presented approach is especially convenient for 3-dimensional and 4 dimensional cases applicable in many engineering and statistical computations, in which significant speed up can be obtained using SSE instructions or GPU use. Also, the presented approach enables symbolic manipulation as the solution of a system of linear equations is transformed to extended cross-product using a matrix form which enables symbolic manipulations.

Direct application of the presented approach has also been demonstrated on the barycentric coordinates computation and simple geometric problems.

The presented approach enables avoiding division operations as a denominator is actually stored in the homogeneous coordinate w. It which leads to significant computational savings, increase of precision and robustness as the division operation is the longest one and the most decreasing precision of computation.

The presented approach also enables derivation of new and more computationally efficient formula in other computational fields.

Acknowledgment. The author would like to thank to colleagues at the University of West Bohemia in Plzen for fruitful discussions and to anonymous reviewers for their comments and hints which helped to improve the manuscript significantly. Special thanks belong also to SIGGRAPH and Eurographics tutorials attendee for their constructive questions, which stimulated this work.

This research was supported by the MSMT CZ - project No. LH12181.

References

1. Coxeter, H.S.M.: Introduction to Geometry. Wiley, New York (1961)
2. Doran, Ch., Lasenby, A.: Geometric Algebra for Physicists. Cambridge University Press, Cambridge (2003)

3. Dorst, L., Fontine, D., Mann, S.: Geometric Algebra for Computer Science. Morgan Kaufmann, San Francisco (2007)
4. Calvet, R.G.: Treatise of Plane Geometry through Geometric Algebra (2007)
5. Hartley, R., Zisserman, A.: MultiView Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
6. Hildenbrand, D.: Foundations of Geometric Algebra Computing. Geometry and Computing. Springer, Heidelberg (2012)
7. Kanatani, K.: Understanding Geometric Algebra. CRC Press, Boca Raton (2015)
8. Krumm, J.: Intersection of Two Planes, Microsoft Research, May 2000. http://research.microsoft.com/apps/pubs/default.aspx?id=68640
9. Johnson, M.: Proof by duality: or the discovery of "new" theorems. Math. Today **32**(11), 171–174 (1996)
10. MacDonald, A.: Linear and Geometric Algebra. CreateSpace, Charleston (2011)
11. Skala, V.: A new approach to line and line segment clipping in homogeneous coordinates. Vis. Comput. **21**(11), 905–914 (2005)
12. Skala, V.: Length, area and volume computation in homogeneous coordinates. Int. J. Image Graph. **6**(4), 625–639 (2006)
13. Skala, V.: Barycentric Coordinates Computation in Homogeneous Coordinates. Comput. Graph. **32**(1), 120–127 (2008). ISSN 0097-8493
14. Skala, V.: Projective geometry, duality and precision of computation in computer graphics, visualization and games. In: Tutorial Eurographics 2013, Girona (2013)
15. Skala, V.: Projective Geometry and Duality for Graphics, Games and Visualization - Course SIGGRAPH Asia 2012, Singapore (2012). ISBN:978-1-4503-1757-3
16. Skala, V.: Intersection computation in projective space using homogeneous coordinates. Int. J. Image Graph. **8**(4), 615–628 (2008)
17. Skala, V.: Modified gaussian elimination without division operations. In: ICNAAM 2013, AIP Conference Proceedings Rhodos, Greece, no. 1558, pp. 1936–1939. AIP Publishing (2013)
18. Vince, J.: Geometric Algebra for Computer Science. Springer, London (2008)
19. Wildberger, N.J.: Divine Proportions: Rational Trigonometry to Universal Geometry. Wild Egg Pty, Sydney (2005)
20. Yamaguchi, F.: Computer Aided Geometric Design: A totally Four Dimensional Approach. Springer, Tokyo (2002)