Robust Barycentric Coordinates Computation of the Closest Point to a Hyperplane in E^n

Vaclav Skala
Department of Computer Science and Engineering
Faculty of Applied Sciences
University of West Bohemia, CZ 306 14 Plzen
Czech Republic
http://www.VaclavSkala.eu

***Abstract*—Barycentric coordinates are well known and used in many applications. They are used for a position computation inside of an (n+1)-sided simplex in an n-dimensional space, i.e. in a triangle in E^2 or in a tetrahedron in E^3. There are some cases when the given point is theoretically on the hyperplane, i.e. on a plane in E^3, but due to numerical imprecision is actually not. Also in some cases we need to compute barycentric coordinates of an n-sided simplex in an n-dimensional space, like barycentric coordinates of a point inside or outside of a triangle in a general position in E^3. In those cases different approaches are taken, mostly unreliable and not robust in general. In this paper reliable and robust computation of barycentric coordinates for n-sided simplex in E^n is described.**

Keywords—computer graphics; computer vision; projective geometry; linear system of equations

I. INTRODUCTION

Barycentric coordinates are widely used in many computational packages and fields including computer graphics and computer vision [3], [8]. They are based on a parametrical formulation of objects and they are closely coupled with linear parametric interpolation methods.

However in some cases barycentric coordinates computation is not reliable and robust enough due to a limited precision of available floating point representations. Also computation of barycentric coordinates of orthogonally projected point onto a hyperplane, i.e. a "flat" subset of dimension (n-1) in n-dimensional space, i.e. onto a plane in the case of E^3. Those two problems are closely related due to numerical problems as well.

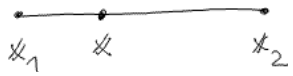
In this paper reliable and robust computation of barycentric coordinates for computing a point position in a simplex or barycentric coordinates computation of a point orthogonally projected to a hyperplane in E^n is described.

II. BARYCENTRIC COORDINATES

Barycentric coordinates are well defined for (n+1)-sided simplexes in n-dimensional space, for a convex hull of n points in (n+1)-dimensional space. In the case of a line segment in E^1 we get a point position inside/outside of the line segment, Fig.1, in the case of a triangle in E^2 it determines a point position in a triangle, Fig.2.a, and in the case of E^3 determines a point position inside of a tetrahedron, see Fig.2.b.

$$\begin{aligned} \lambda_1 x_1 + \lambda_2 x_2 &= x \\ \lambda_1 + \lambda_2 &= 1 \end{aligned} \tag{1}$$

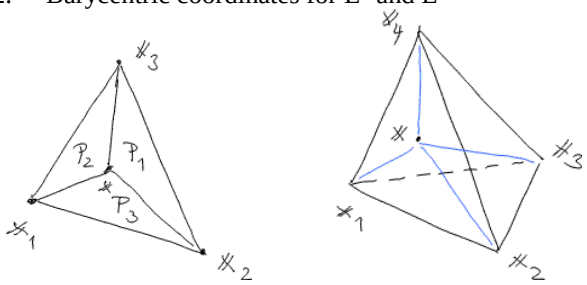
Fig.1. Barycentric coordinates for a line segment



Barycentric coordinates for E^2 are defined as:

$$\begin{aligned} \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 &= x \\ \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 &= y \\ \lambda_1 + \lambda_2 + \lambda_3 &= 1 \end{aligned} \tag{2}$$

Fig.2. Barycentric coordinates for E^2 and E^3



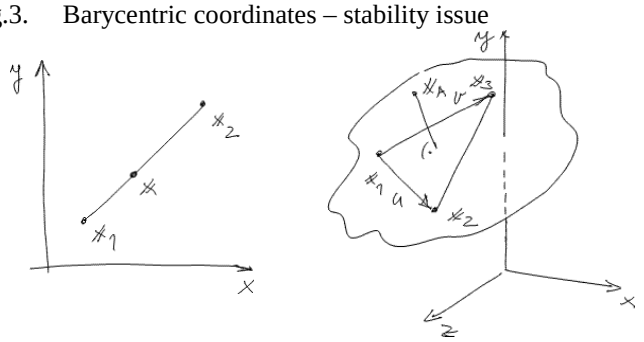
Barycentric coordinates for E^3 are defined as:

$$\begin{aligned} \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4 &= x \\ \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 + \lambda_4 y_4 &= y \\ \lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3 + \lambda_4 z_4 &= z \\ \lambda_1 + \lambda_2 + \lambda_3 &= 1 \end{aligned} \tag{3}$$

Barycentric coordinates have very interesting and properties [20]. In the case of E^1 barycentric coordinates give us ratio of lengths, in the case of E^2 they give us a ration of areas and in the case of E^3 they give us ratio of volumes. Those properties

can be used for reliable computation of intersections, e.g. ray-triangle etc. Barycentric coordinates can be also used for simplexes in n-dimensional space, e.g. for determining a position inside of a convex polygon or polyhedral etc. It should be noted that the barycentric coordinates are also used for point determination for convex n-sided polygon in E^2 [8],[9].

Fig.3. Barycentric coordinates – stability issue



However there is a significant problem if an n-dimensional simplex is given in n-dimensional space, e.g. a line is E^2 or a triangle in E^3 is given, in a general position. Let us consider the simplest case, i.e. barycentric coordinates of a line segment in E^2, see Fig.3.a and Fig.3.b.

In this case computation of the barycentric coordinates would lead to a solution of linear equations.

$$\begin{aligned} \lambda_1 x_1 + \lambda_2 x_2 &= x \\ \lambda_1 y_1 + \lambda_2 y_2 &= y \\ \lambda_1 + \lambda_2 &= 1 \end{aligned} \tag{4}$$

This is perfectly valid formulation if unlimited precision is used. However if $x_2 - x_1 = 0$ or $y_2 - y_1 = 0$ the system is singular and one of those equations hast to be taken and the condition $\lambda_1 + \lambda_2 = 1$ has to be used. Unfortunately available floating point representation has a limited mantissa length and even quadruple or extended length is not supported in many programming tools. This aspect, when if $x_2 - x_1 \to 0$ or $y_2 - y_1 \to 0$, results into non-reliable computation of barycentric coordinates in principle.

It our case E^2 it means that the point (x, y) is not actually on the line, but is "aside" of the given line due to a limited precision. Programmers usually find some "engineering" solution to handle this problem, but the code is not "clear" and becoming complicated in general. Usual approaches are:

- Select a coordinate with the biggest distance – there must be several "**IF**" statements in an n-dimensional case
- Rotate the simplex to some basic position – difficult to find general rotation to make in n-dimensional case

However all those approaches are not robust and simple to implement correctly in the n-dimensional case. In the following a robust solution for the n-dimensional case will be presented.

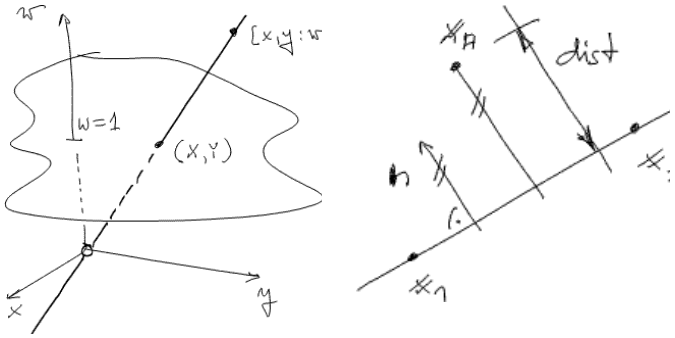
III. PROJECTIVE SPACE AND DUALITY

Projective extension of the Euclidean space is usually considered only as a "tool" useful for geometric transformations in computer graphics and computer vision. A point $(X, Y) \in E^2$ is represented in projective space by homogeneous coordinates $\boldsymbol{x} = [x, y: w]^T$, where: w is the homogeneous coordinate [1], [6], [9]. Fig.4.a presents a geometric interpretation. It can be seen that transformation to the Euclidean space is given as

$$X = {}^{x}/_{w} \qquad Y = {}^{y}/_{w} \tag{5}$$

and $w \neq 0$.

Fig.4. Duality and distance



In the case of E^2 a line is given in the implicit form as

$$aX + bY + d = 0 \tag{6}$$

If this equation is multiplied by $w \neq 0$ we get

$$awX + bwY + wd = 0 \tag{7}$$

i.e.

$$ax + by + dw = 0 \tag{8}$$

If the vector notation is used

$$\boldsymbol{a}^T \boldsymbol{x} = 0 \tag{9}$$

where $\boldsymbol{a} = [a, b: d]^T$ and $\boldsymbol{x} = [x, y: w]^T$. It should be noted that $\boldsymbol{n} = [a, b]^T$ is a normal vector of a line, i.e. it is a direction, while d relates to a distance of the line from the origin.

The distance of a point $\boldsymbol{\xi} = [\xi_x, \xi_y: \xi_w]^T$ from a line can be computed as

$$dist = \frac{a\xi_x + b\xi_y + d\xi_w}{\xi_w \sqrt{a^2 + b^2}} \tag{10}$$

In the case of E^3 a plane is determined as

$$ax + by + cz + dw = 0 \tag{11}$$

i.e.

$$\boldsymbol{a}^T \boldsymbol{x} = 0 \tag{12}$$

where $\boldsymbol{a} = [a, b, c: d]^T$, $\boldsymbol{x} = [x, y, z: w]^T$ and a normal vector of the plane is defined as $\boldsymbol{n} = [a, b, c]^T$. From the description above it can be seen that meaning of symbols can be

exchanged and the equation describes the same geometric object, i.e. a line in this case.

This is due to the principle of duality [2], [5], [7], [10] which says that

- in the E^2 case a point is dual to a line and vice versa
- in the E^3 case a point is dual to a plane and vice versa

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 [5], [10].

It can be shown that computation of an intersection of two lines in E^2 is dual to computation of a line given by two points (it is actually join operation). It means that there should be same programming sequence for solving both dual cases, i.e.

$$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2 \tag{13}$$

i.e.

$$det\begin{bmatrix} x & y & w \\ a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \end{bmatrix} = 0 \tag{14}$$

and

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \tag{15}$$

i.e.

$$det\begin{bmatrix} a & b & d \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0 \tag{16}$$

In the case of E^3 a plane given as a join of three points and dual problem, i.e. an intersection of three planes, can be computed as

$$\boldsymbol{x} = \boldsymbol{\varrho}_1 \times \boldsymbol{\varrho}_2 \times \boldsymbol{\varrho}_3 \tag{17}$$

i.e.

$$det\begin{bmatrix} x & y & z & w \\ a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{bmatrix} = 0 \tag{18}$$

and

$$\boldsymbol{\varrho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3 \tag{19}$$

i.e.

$$det\begin{bmatrix} a & b & c & d \\ x_1 & y_1 & z_1 & w_1 \\ x_2 & y_2 & z_2 & w_2 \\ x_3 & y_3 & z_3 & w_3 \end{bmatrix} = 0 \tag{20}$$

There are significant advantages of this approach:

- Natural support for GPU computation leading to significant speed up.
- There is no division operation used that is needed if an intersection is computed in the Euclidean space, like $det_x/_{det}$, which leads to instability in principle.
- There are no special cases which a programmer has to take care of, i.e. detection of collinearity etc.
- If points are given in homogeneous coordinates, transformation to the Euclidean coordinates is not required. It means that 6, resp. 9 division operations are not needed in the case of E^2, resp. E^3 and higher precision can be expected as well.

It can be proved that solution of a linear system of equations is equivalent to an extended cross-product [9]. This is very significant result as instead of solving a linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ we can use extended cross-product can be used. It should be noted that replacing of a solution of linear system of equations $\boldsymbol{Ax} = \boldsymbol{b}$ by extended cross-product $\boldsymbol{x} = \boldsymbol{\xi}_1 \times \boldsymbol{\xi}_2 \times ... \times \boldsymbol{\xi}_n$ enables further formal manipulation using linear algebra.

As a typical example of this approach is computation of barycentric coordinates which can be used also for the case, when points $x_1, \ldots, x_n$, are given in homogeneous coordinates in E^N, in general. However it is useful to remind how barycentric coordinates can be computed using projective representation, see [7], [8] for detailed description.

Barycentric coordinates

E^1 case

Let us consider vectors

$$\begin{aligned} \boldsymbol{x} &= [x_1, x_2 : x]^T \\ \boldsymbol{w} &= [w_1, w_2 : w]^T \end{aligned} \tag{21}$$

Then barycentric coordinates are given as

$$\boldsymbol{\xi} = \boldsymbol{x} \times \boldsymbol{w} = [\xi_1, \xi_2 : \xi_w]^T \tag{22}$$

The Euclidean barycentric coordinates are given as:

$$\lambda_1 = -\frac{\xi_1}{\xi_w} \quad \lambda_2 = -\frac{\xi_2}{\xi_w} \tag{23}$$

E^2 case

Let us consider vectors

$$\begin{aligned} \boldsymbol{x} &= [x_1, x_2, x_3 : x]^T \\ \boldsymbol{y} &= [y_1, y_2, y_3 : y]^T \\ \boldsymbol{w} &= [w_1, w_2, w_3 : w]^T \end{aligned} \tag{24}$$

Then projective barycentric coordinates are given as

$$\boldsymbol{\xi} = \boldsymbol{x} \times \boldsymbol{y} \times \boldsymbol{w} = [\xi_1, \xi_2, \xi_3 : \xi_w]^T \tag{25}$$

The Euclidean barycentric coordinates are given as:

$$\lambda_1 = -\frac{\xi_1}{\xi_w} \quad \lambda_2 = -\frac{\xi_2}{\xi_w} \quad \lambda_3 = -\frac{\xi_3}{\xi_w} \tag{26}$$

E^3 case

Let us consider vectors

$$\begin{aligned} \boldsymbol{x} &= [x_1, x_2, x_3, x_4 : x]^T \\ \boldsymbol{y} &= [y_1, y_2, y_3, y_4 : y]^T \\ \boldsymbol{z} &= [z_1, z_2, z_3, z_4 : z]^T \\ \boldsymbol{w} &= [w_1, w_2, w_3, w_4 : w]^T \end{aligned} \tag{27}$$

Then projective barycentric coordinates are given as

$$\boldsymbol{\xi} = \boldsymbol{x} \times \boldsymbol{y} \times \boldsymbol{z} \times \boldsymbol{w} = [\xi_1, \xi_2, \xi_3, \xi_4 : \xi_w]^T \tag{28}$$

The Euclidean barycentric coordinates are given as:

$$\lambda_1 = -\frac{\xi_1}{\xi_w} \quad \lambda_2 = -\frac{\xi_2}{\xi_w} \quad \lambda_3 = -\frac{\xi_3}{\xi_w} \quad \lambda_4 = -\frac{\xi_4}{\xi_w} \tag{29}$$

How simple and elegant solutions!

As due to imprecision of computation a point never lies at the hyperplane, i.e. on a plane in the case of E^3, we have actually a problem, how to find barycentric coordinates of orthogonally projected point to a hyperplane, i.e. onto plane on which a triangle lies. It means that there is a question how to compute:

- barycentric coordinates of a point in E^N of an n-sided simplex
- barycentric coordinates of an orthogonally projected point in E^N to an n-sided simplex

Let us consider computation of barycentric coordinates of a line in E^2. As recently shown the barycentric coordinates in E^2 are computed as

$$\begin{gathered} \lambda_1 x_1 + \lambda_2 x_2 = x \\ \lambda_1 y_1 + \lambda_2 y_2 = y \\ \lambda_1 + \lambda_2 = 1 \end{gathered} \tag{30}$$

It can be seen that x and y coordinates are bounded by the equation $ax + by + d = 0$ so the system of linear equations is redundant.

IV. Robust Computation of Barycentric Coordinates

Robustness of computation is a key issue in many sophisticated computational algorithms as sophisticated engineering problems solved today might be ill conditioned. Let us explore how barycentric coordinates of an orthogonally projected point to an n-sided simplex in n-dimensional space can be reliably computed, i.e. barycentric coordinates of a point orthogonally projected to a plane on which a triangle lies.

For the sake of simplicity, the case of E^2 will be used without any loss of generality for the approach explanation.

E^2 case

Let us define a distance of the point $\boldsymbol{x}_A \in E^2$ from n-sided simplex in n-dimensional space, i.e. a line segment in a general position in E^2, Fig.4.a, as

$$dist^2 = [\boldsymbol{x}(t) - \boldsymbol{x}_A]^T [\boldsymbol{x}(t) - \boldsymbol{x}_A] \tag{31}$$

A line on which the line segment $\boldsymbol{x}_1\boldsymbol{x}_2$ lies is defined as

$$\boldsymbol{x}(t) = \boldsymbol{x}_1 + \boldsymbol{s}t = \boldsymbol{x}_1 + (\boldsymbol{x}_2 - \boldsymbol{x}_1)t \tag{32}$$

We want to know a point on a line, which is the closest point to the given point $\boldsymbol{x}_A$, i.e. a value of the parameter t, and for this point to find its barycentric coordinates. Then for finding an extreme the following condition has to be solved

$$\begin{gathered} \frac{d}{dt}(dist^2) = \\ \frac{d}{dt}\{[\boldsymbol{x}_1 + \boldsymbol{s}t - \boldsymbol{x}_A]^T [\boldsymbol{x}_1 + \boldsymbol{s}t - \boldsymbol{x}_A]\} = \\ 2[\boldsymbol{x}_1 + \boldsymbol{s}t - \boldsymbol{x}_A]^T \boldsymbol{s} = \\ 2\boldsymbol{s}^T \boldsymbol{s}\, t + 2\boldsymbol{s}^T(\boldsymbol{x}_1 - \boldsymbol{x}_A) = \\ \boldsymbol{s}^T \boldsymbol{s}\, t + \boldsymbol{s}^T(\boldsymbol{x}_1 - \boldsymbol{x}_A) = 0 \end{gathered} \tag{33}$$

then

$$t_0 = -\frac{\boldsymbol{s}^T(\boldsymbol{x}_1 - \boldsymbol{x}_A)}{\boldsymbol{s}^T \boldsymbol{s}} \tag{34}$$

or as homogeneous scalar value

$$t_0 = [\boldsymbol{s}^T(\boldsymbol{x}_A - \boldsymbol{x}_1) : \boldsymbol{s}^T \boldsymbol{s}]^T \tag{35}$$

As we know that

$$\begin{gathered} \boldsymbol{x}(t_0) = \boldsymbol{x}_1 + \boldsymbol{s}t_0 = \boldsymbol{x}_1 + (\boldsymbol{x}_2 - \boldsymbol{x}_1)t_0 \\ \boldsymbol{x}(\lambda_1, \lambda_2) = \lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 \end{gathered} \tag{36}$$

Therefore $\lambda_1 = 1 - t_0$ and $\lambda_2 = t_0$.

Formulation using points instead for vectors

The barycentric coordinates can be computed directly as follows

$$dist^2 = [\boldsymbol{x}(\lambda_1, \lambda_2) - \boldsymbol{x}_A]^T [\boldsymbol{x}(\lambda_1, \lambda_2) - \boldsymbol{x}_A] \tag{37}$$

$$\begin{gathered} \boldsymbol{x}(\lambda_1, \lambda_2) = \lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_1 \\ \lambda_1 + \lambda_2 = 1 \end{gathered} \tag{38}$$

Then

$$\begin{gathered} \frac{\partial}{\partial \lambda_1} dist^2 = \\ \frac{\partial}{\partial \lambda_1}\{[\lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 - \boldsymbol{x}_A]^T [\lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 - \boldsymbol{x}_A]\} \\ = 2[\lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 - \boldsymbol{x}_A]^T \boldsymbol{x}_1 = 0 \end{gathered} \tag{39}$$

and

$$\begin{gathered} \frac{\partial}{\partial \lambda_2} dist^2 = \\ \frac{\partial}{\partial \lambda_2}\{[\lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 - \boldsymbol{x}_A]^T [\lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 - \boldsymbol{x}_A]\} \\ = 2[\lambda_1 \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_2 - \boldsymbol{x}_A]^T \boldsymbol{x}_2 = 0 \end{gathered} \tag{40}$$

Therefore

$$\begin{gathered} \lambda_1 \boldsymbol{x}_1^T \boldsymbol{x}_1 + \lambda_2 \boldsymbol{x}_1^T \boldsymbol{x}_2 - \boldsymbol{x}_1^T \boldsymbol{x}_A = 0 \\ \lambda_1 \boldsymbol{x}_1^T \boldsymbol{x}_2 + \lambda_2 \boldsymbol{x}_2^T \boldsymbol{x}_2 - \boldsymbol{x}_2^T \boldsymbol{x}_A = 0 \end{gathered} \tag{41}$$

or in the matrix form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ as

$$\begin{bmatrix} \boldsymbol{x}_1^T \boldsymbol{x}_1 & \boldsymbol{x}_1^T \boldsymbol{x}_2 \\ \boldsymbol{x}_1^T \boldsymbol{x}_2 & \boldsymbol{x}_2^T \boldsymbol{x}_2 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^T \boldsymbol{x}_A \\ \boldsymbol{x}_2^T \boldsymbol{x}_A \end{bmatrix} \tag{42}$$

Solving this system of linear equations we get values of λ_1 and λ_2.

Stability issues

Stability is given by det value, which is given as

$$det(\boldsymbol{A}) = det\begin{bmatrix} \boldsymbol{x}_1^T\boldsymbol{x}_1 & \boldsymbol{x}_1^T\boldsymbol{x}_2 \\ \boldsymbol{x}_1^T\boldsymbol{x}_2 & \boldsymbol{x}_2^T\boldsymbol{x}_2 \end{bmatrix} = \boldsymbol{x}_1^T\boldsymbol{x}_1 \,.\, \boldsymbol{x}_2^T\boldsymbol{x}_2 - (\boldsymbol{x}_1^T\boldsymbol{x}_2)^2 \tag{43}$$

i.e.

$$\begin{aligned} det(\boldsymbol{A}) &= d_1^2 d_2^2 - (d_1 d_2)^2 \cos^2(\varphi) \\ &= d_1^2 d_2^2 (1 - \cos^2(\varphi)) \\ &= d_1^2 d_2^2 \sin^2(\varphi) \end{aligned} \tag{44}$$

If $\boldsymbol{x}_1 = \boldsymbol{0}$, i.e. the point is in the origin, then $det(\boldsymbol{A}) = 0$. It is a special case and therefore

$$\lambda_2 \boldsymbol{x}_2^T\boldsymbol{x}_2 - \boldsymbol{x}_2^T\boldsymbol{x}_A = 0 \tag{45}$$

and

$$\lambda_2 = -\frac{\boldsymbol{x}_2^T\boldsymbol{x}_A}{\boldsymbol{x}_2^T\boldsymbol{x}_2} \qquad \lambda_1 = 1 - \lambda_2 \tag{46}$$

It can be seen that this approach is not reliable and robust.

Of course, there is another possibility, how to compute barycentric coordinates of a projected point, i.e. a line passing the given point with a directional vector equal to a normal of the line on which the line segment lies and compute parameters t for the line $\boldsymbol{x}(t)$ and then from this parameter the barycentric coordinates. However this approach has division operations used. This approach can be described as follows.

A line given by two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$

$$\boldsymbol{n}^T\boldsymbol{x} + d = 0 \tag{47}$$

and a ray from the point $\boldsymbol{x}_A$ with the directional vector $\boldsymbol{s}$

$$\boldsymbol{x}(t) = \boldsymbol{x}_A + \boldsymbol{s}\, t \tag{48}$$

We get a parameter t_0 for an intersection point as

$$\boldsymbol{n}^T\boldsymbol{x}_A + \boldsymbol{n}^T\boldsymbol{s}\, t_0 + d = 0 \tag{49}$$

then

$$t_0 = -\frac{\boldsymbol{n}^T\boldsymbol{x}_A + d}{\boldsymbol{n}^T\boldsymbol{s}} \tag{50}$$

Then the intersection point $\boldsymbol{x}(t_0)$ is

$$\boldsymbol{x}(t) = \boldsymbol{x}_A - \boldsymbol{s}\frac{\boldsymbol{n}^T\boldsymbol{x}_A + d}{\boldsymbol{n}^T\boldsymbol{s}} \tag{51}$$

As the normal vector $\boldsymbol{n}$ of the line is identical with the directional vector of a ray the intersection point is given as

$$\begin{aligned} \boldsymbol{x}(t_0) &= \boldsymbol{x}_A - \boldsymbol{n}\frac{\boldsymbol{n}^T\boldsymbol{x}_A + d}{\boldsymbol{n}^T\boldsymbol{n}} \\ &= \boldsymbol{x}_A - \frac{\boldsymbol{n}}{\boldsymbol{n}^T\boldsymbol{n}}(\boldsymbol{n}^T\boldsymbol{x}_A + d) \end{aligned} \tag{52}$$

and then $\lambda_1 = 1 - t_0$ and $\lambda_2 = t_0$. It is necessary to note, that points $\boldsymbol{x}$ have to be given in the Euclidean space and if given in projective coordinates additional division operation has to be used. Simpler formula can be obtained if projective space is used.

E^3 case

The barycentric coordinates for a triangle in a general position in E^3, Fig. 3b,can be determined similarly in the case of E^2 as follows

$$dist^2 = [\boldsymbol{x}(u,v) - \boldsymbol{x}_A]^T[\boldsymbol{x}(u,v) - \boldsymbol{x}_A] \tag{53}$$

where

$$\boldsymbol{x}(u,v) = \boldsymbol{x}_1 + \boldsymbol{s}_1 u + \boldsymbol{s}_2 v \tag{54}$$

is a plane on which the triangle $\boldsymbol{x}_1\boldsymbol{x}_2\boldsymbol{x}_3$ lies. We want to know a point on a plane, which is the closest point to the given point $\boldsymbol{x}_A$, i.e. a value of the parameters u, v.

Then the following conditions for finding an extreme must be solved

$$\frac{\partial}{\partial u}(dist^2) = 0 \tag{55}$$

and

$$\frac{\partial}{\partial v}(dist^2) = 0 \tag{56}$$

i.e.

$$\begin{aligned} &\frac{\partial}{\partial u}(dist^2) = \frac{\partial}{\partial u}\{[\boldsymbol{x}(u,v) - \boldsymbol{x}_A]^T[\boldsymbol{x}(u,v) - \boldsymbol{x}_A]\} = \\ &\frac{\partial}{\partial u}\{[\boldsymbol{x}_1 + \boldsymbol{s}_1 u + \boldsymbol{s}_2 v - \boldsymbol{x}_A]^T[\boldsymbol{x}_1 + \boldsymbol{s}_1 u + \boldsymbol{s}_2 v - \boldsymbol{x}_A]\} \\ &\quad = 2[\boldsymbol{x}_1 + \boldsymbol{s}_1 u + \boldsymbol{s}_2 v - \boldsymbol{x}_A]^T\boldsymbol{s}_1 \\ &\quad = \boldsymbol{s}_1^T\boldsymbol{s}_1 u + \boldsymbol{s}_1^T\boldsymbol{s}_2 v + \boldsymbol{s}_1^T(\boldsymbol{x}_1 - \boldsymbol{x}_A) \\ &\quad = 0 \end{aligned} \tag{57}$$

and

$$\begin{aligned} &\frac{\partial}{\partial v}(dist^2) = \frac{\partial}{\partial v}\{[\boldsymbol{x}(u,v) - \boldsymbol{x}_A]^T[\boldsymbol{x}(u,v) - \boldsymbol{x}]\} = \\ &\frac{\partial}{\partial v}\{[\boldsymbol{x}_1 + \boldsymbol{s}_1 u + \boldsymbol{s}_2 v - \boldsymbol{x}_A]^T[\boldsymbol{x}_1 + \boldsymbol{s}_1 u + \boldsymbol{s}_2 v - \boldsymbol{x}_A]\} \\ &\quad = 2[\boldsymbol{x}_1 + \boldsymbol{s}_1 u + \boldsymbol{s}_2 v - \boldsymbol{x}_A]^T\boldsymbol{s}_2 \\ &\quad = \boldsymbol{s}_1^T\boldsymbol{s}_2 u + \boldsymbol{s}_2^T\boldsymbol{s}_2 v + \boldsymbol{s}_1^T\boldsymbol{s}_2 u_2^T\boldsymbol{s}_2 v \\ &\quad + \boldsymbol{s}_2^T(\boldsymbol{x}_1 - \boldsymbol{x}_A) = 0 \end{aligned} \tag{58}$$

It means that a linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has to be solved, i.e.

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{s}_1^T\boldsymbol{s}_1 & \boldsymbol{s}_1^T\boldsymbol{s}_2 \\ \boldsymbol{s}_1^T\boldsymbol{s}_2 & \boldsymbol{s}_2^T\boldsymbol{s}_2 \end{bmatrix} \quad \boldsymbol{x} = \begin{bmatrix} u \\ v \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{s}_1^T(\boldsymbol{x}_1 - \boldsymbol{x}_A) \\ \boldsymbol{s}_2^T(\boldsymbol{x}_1 - \boldsymbol{x}_A) \end{bmatrix} \tag{59}$$

The system can be reformulated for a projective solution which more convenient for GPU use if extended cross-product is used as follows

$$\begin{aligned}\boldsymbol{\xi} &= [\boldsymbol{s}_1^T\boldsymbol{s}_1, \boldsymbol{s}_1^T\boldsymbol{s}_2 : -\boldsymbol{s}_1^T(\boldsymbol{x}_1 - \boldsymbol{x}_A)]^T \\ \boldsymbol{\eta} &= [\boldsymbol{s}_1^T\boldsymbol{s}_2, \boldsymbol{s}_2^T\boldsymbol{s}_2 : -\boldsymbol{s}_2^T(\boldsymbol{x}_1 - \boldsymbol{x}_A)]^T\end{aligned} \tag{60}$$

Then parametric coordinates of a point $\boldsymbol{x}$, which is an orthogonal projection of the points $\boldsymbol{x}_A$ onto a plane, can be computed as

$$\boldsymbol{\sigma} = \boldsymbol{\xi} \times \boldsymbol{\eta} \tag{61}$$

where: $\boldsymbol{\sigma} = [\sigma_1, \sigma_2 : \sigma_w]^T$ are parameters coordinates (u, v) using homogeneous coordinates. The parametric coordinates of a point in the Euclidean space are then given as $u = \frac{\sigma_1}{\sigma_w}$ and $v = \frac{\sigma_2}{\sigma_w}$.

As we know that

$$\begin{aligned}\boldsymbol{x}(u,v) &= \boldsymbol{x}_1 + \boldsymbol{s}_1 u + \boldsymbol{s}_2 u \\ &= \boldsymbol{x}_1 + (\boldsymbol{x}_2 - \boldsymbol{x}_1)u \\ &\quad + (\boldsymbol{x}_3 - \boldsymbol{x}_1)v\end{aligned} \tag{62}$$

and

$$\boldsymbol{x}(\lambda_1, \lambda_2, \lambda_3) = \lambda_1\boldsymbol{x}_1 + \lambda_2\boldsymbol{x}_2 + \lambda_3\boldsymbol{x}_3 \tag{63}$$

Therefore $\lambda_1 = 1 - u - v$ and $\lambda_2 = u$ and $\lambda_3 = v$.

Stability issues

Stability is given by det value, which is given as

$$\begin{aligned}det(\boldsymbol{A}) &= det\begin{bmatrix}\boldsymbol{s}_1^T\boldsymbol{s}_1 & \boldsymbol{s}_1^T\boldsymbol{s}_2 \\ \boldsymbol{s}_1^T\boldsymbol{s}_2 & \boldsymbol{s}_2^T\boldsymbol{s}_2\end{bmatrix} \\ &= \boldsymbol{s}_1^T\boldsymbol{s}_1\ \boldsymbol{s}_2^T\boldsymbol{s}_2 - (\boldsymbol{s}_1^T\boldsymbol{s}_2)^2 \\ &= \boldsymbol{s}_1^2\boldsymbol{s}_2^2 - (\boldsymbol{s}_1\boldsymbol{s}_2)^2\cos^2(\varphi) \\ &= \boldsymbol{s}_1^2\boldsymbol{s}_2^2(1 - \cos^2(\varphi)) \\ &= \boldsymbol{s}_1^2\boldsymbol{s}_2^2\sin^2(\varphi)\end{aligned} \tag{64}$$

where φ is an angle between vectors $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. As

$$\boldsymbol{s}_1^T\boldsymbol{s}_2 = \|\mathbf{s}_1\|\|\mathbf{s}_2\|\sin(\varphi) \tag{65}$$

and

$$\|\mathbf{s}_1\|^2 = \boldsymbol{s}_1^2 \quad \|\mathbf{s}_2\|^2 = \boldsymbol{s}_2^2 \tag{66}$$

It means that the value $det(\boldsymbol{A}) \to 0$ if a triangle is getting very slim. Formulation with points is not considered as it is instable.

Of course, there is another possibility, how to compute barycentric coordinates of the projected point, i.e. a line passing the given point with a directional vector equal to a normal of the plane on which the triangle lies and compute parameters u, v for the plane $\boldsymbol{x}(u, v)$ and then from those parameters the barycentric coordinates. However this approach has several division operations used.

V. Conclusion

A robust method for barycentric coordinates computation of a point and n-sided simplex in n-dimensional space, e.g. a point and a triangle in E^3, is presented in this paper. The presented approach is based on optimization formulation in E^N. It can be used for barycentric coordinates computations of orthogonally projected points onto a hyperplane if the point does not lie on the hyperplane, Fig.4. Theoretical analysis of robustness and experiments made proved significantly better efficiency and precision if a vector notation is used instead of points.

Acknowledgment

The author would like to thank to anonymous reviewers for their critical comments and hints that improved this paper significantly. Thanks also belong to students and colleagues at the University of West Bohemia, Plzen and VSB-Technical University, Ostrava for their suggestions. The project was supported by the Ministry of Education of the Czech Republic, projects No.LG13047 and LH12181.

References

[1] Bloomenthal,J., Rokne,J.: Homogeneous Coordinates, The Visual Computer, Vol.11,No.1, pp.15-26, 1994.

[2] Coxeter,H.S.M.: Introduction to Geometry, John Wiley, 1969.

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

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

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

[6] Skala,V.: A New Approach to Line and Line Segment Clipping in Homogeneous Coordinates, The Visual Computer, Vol.21, No.11, pp.905 914, Springer Verlag, 2005

[7] Skala,V.: Length, Area and Volume Computation in Homogeneous Coordinates, International Journal of Image and Graphics, Vol.6., No.4, pp.625-639, 2006

[8] Skala,V.: Barycentric Coordinates Computation in Homogeneous Coordinates, Computers & Graphics, Elsevier, ISSN 0097-8493, Vol.32, No.1, pp.120-127, 2008

[9] Skala,V.: Intersection Computation in Projective Space using Homogeneous Coordinates, Int.Journal on Image and Graphics, ISSN 0219-4678, Vol.8, No.4, pp.615-628, 2008

[10] Skala,V.: Projective Geometry and Duality for Graphics, Games and Visualization - Course SIGGRAPH Asia 2012, Singapore, ISBN 978-1-4503-1757-3, 2012 .