PROCEEDINGS
1st International Workshop on Line Geometry and Kinematics (IW-LGK-11) pages 105-111
April 26-30, 2011, Paphos Cyprus, pp. XX-XX
ISBN: 978-3-86780-244-4

GEOMETRIC COMPUTATION, DUALITY AND PROJECTIVE SPACE

Vaclav SKALA
University of West Bohemia, Plzen, Czech Republic

ABSTRACT: This paper presents solution of some selected problems that can be easily solved in the projective space. Projective space and homogeneous coordinates are mostly used in computer graphics and used especially for geometric transformations. Nevertheless the projective formulation offers an elegant solution to several geometrical problems, too. If the principle of duality is used, quite surprising solutions can be found and new useful theorems can be generated as well, e.g. an equation of a line in E^3 as an intersection of two planes, computation of barycentric coordinates etc. The projective extension of the Euclidean space also offers higher computational stability as it avoids division operation in general. In this paper we will show that a solution of a system of linear equations is equivalent to the extended cross (outer) product and how this can be used for solving geometrical problems. As the duality between points and lines in E^2 or between points and planes in E^3 is known, many geometrical algorithms can be easily converted to the dual representation, solved in the dual space and the result can be converted back to the projective or Euclidean space, e.g. point-in-polygon test is dual to a test if a line intersect a polygon. The presented approach offers higher robustness and it is convenient for matrix-vector supporting hardware namely for the GPU implementation.

Keywords: Projective geometry, Duality, Intersection, Barycentric coordinates, Plücker coordinates.

1. INTRODUCTION

Homogeneous coordinates are mostly used in computer graphics for representation of geometric transformations like translation, rotation etc. and for projection operations. In many cases homogeneous coordinates are only seen as a "mathematical tool" that makes a simple description of geometric transformations possible. Nevertheless there are many invisible impacts on the algorithm design that leads to new, faster and robust algorithms convenient for GPU implementation.

The principle of duality is not usually explored at the technical universities, but it helps a lot in development of new algorithms [1], especially related to computer graphics, computer vision and geometry as well. Let us consider a simple case in E^2, e.g. a line and a point. It is known that a line and a point are dual. It means that $\boldsymbol{A}^T\boldsymbol{X} = 0$ is valid for all points $\boldsymbol{X} = [X, Y: 1]^T$ of the line given by the vector $\boldsymbol{A} = [A, B: C]^T$, i.e. satisfying $AX + BY + C = 0$. Because of the principle of duality the meaning of $\boldsymbol{X}$ and $\boldsymbol{A}$ symbols can be exchanged. As the equation $\boldsymbol{A}^T\boldsymbol{X} = 0$ is implicit, it can be multiplied by any $w \neq 0$ and the geometrical entity remains. It means that a line p is defined by the equation $\boldsymbol{a}^T\boldsymbol{x} = 0$, where, $\boldsymbol{x} = [x, y: w]^T$, $X = \frac{x}{w}$ and $Y = \frac{y}{w}$ [6]. A line p given by two points can be determined as $\boldsymbol{a} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$ and due to the principle of duality, we can determine an intersection of two lines as $\boldsymbol{x} = \boldsymbol{a}_1 \times \boldsymbol{a}_2$.

$$\boldsymbol{a} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 = det\begin{pmatrix} a & b & c \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{pmatrix}$$

Similarly for a plane ρ computation given by three points we can write $\boldsymbol{\rho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3$ and using principle of duality an intersection pintt x of three planes is given as $\boldsymbol{x} = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3$.

We can see that "normalized" vectors are usually $\overline{\boldsymbol{a}} = \frac{[a,b:c]^T}{\sqrt{a^2+b^2}}$ and $\overline{\boldsymbol{x}} = \frac{[x,y:w]^T}{w}$. The distance d of the point $\boldsymbol{x}$ from the line p can be easily expressed as $d = (ax + by + cw)/\left(w\sqrt{a^2 + b^2}\right)$.

In some cases we do need "normalized" vectors for computation, but not in all. Let us consider well known barycentric coordinates.

2. GEOMETRIC TRANSFORMATIONS

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 line $\boldsymbol{a}'$, if geometric transformation $\boldsymbol{T}$ is applied on points.

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

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

For the standard geometric transformations rotation and translation $\det(\boldsymbol{T}) = 1$ and 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 projective notation as

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

where: $\cos(\varphi) = a/c$ and $\sin(\varphi) = b/c$. In this case $\det(\boldsymbol{T}) \neq 1$ of course.

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 solve a problem: Given a line *p* and a geometric transformation **T**. How the coefficients of a line are changed? Similarly for a plane in E^3 and dual problems a solution is simple.

3. PLÜCKER COORDINATES

Plücker coordinates are not known in computer graphics community. Nevertheless they offer a nice tool especially in the context of duality. The Plücker coordinates are used for a computation of a line in E^3 in the parametric form if given by two points in homogeneous coordinates. Due to the principle of duality we can easily derive a solution for an intersection of two as points and planes are dual in E^3.

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

$$\mathbf{q}(t) = \frac{\mathbf{v}\times\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2} + \boldsymbol{\omega}t \qquad \text{or} \qquad \text{as} \quad \hat{\mathbf{q}}(t) = \left[\mathbf{v}\times\boldsymbol{\omega} + t\boldsymbol{\omega}\|\boldsymbol{\omega}\|^2 : \|\boldsymbol{\omega}\|^2\right] \text{ if projective notation is used.}$$

Let us consider two points in the homogeneous coordinates: $\boldsymbol{x}_1 = [x_1, y_1, z_1, w_1]^T$, $\boldsymbol{x}_2 = [x_2, y_2, z_2, w_2]^T$. The Plücker coordinates l_{ij} are defined as follows:

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

$$l_{41} = w_1x_2 - w_2x_1 \qquad l_{23} = y_1z_2 - y_2z_1 \qquad \boldsymbol{\omega} = [l_{41}, l_{42}, l_{43}]^T$$
$$l_{42} = w_1y_2 - w_2y_1 \qquad l_{31} = z_1x_2 - z_2x_1 \qquad \mathbf{v} = [l_{23}, l_{31}, l_{12}]^T$$
$$l_{43} = w_1z_2 - w_2z_1 \qquad l_{12} = x_1y_2 - x_2y_1$$

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

Two vectors **ω** and **v** are defined as $\mathbf{X}_2 - \mathbf{X}_1 = \boldsymbol{\omega}$, $\mathbf{X}_1 \times \mathbf{X}_2 = \mathbf{v}$,where: $\mathbf{X}_i = [x_i, y_i, z_i]^T/w_i$ are points in the Euclidean space and for a general case $w_i \neq 1$ when $\boldsymbol{x}_i$ are not ideal points, i.e. $w_i \neq 0$, we get:

$$\boldsymbol{\omega} = w_2 w_1 (\mathbf{X}_2 - \mathbf{X}_1) = (x_2 w_1 - x_1 w_2 \ , \ y_2 w_1 - y_1 w_2 \ , \ z_2 w_1 - z_1 w_2) = (l_{41}, l_{42}, l_{43})$$

$$\mathbf{v} = w_2 w_1 (\mathbf{X}_1 \times \mathbf{X}_2) = (y_1 z_2 - y_2 z_1 \ , \ z_1 x_2 - z_2 x_1 \ , \ x_1 y_2 - y_1 x_2) = (l_{23}, \ l_{31}, l_{12})$$

It means that **ω** represents the "directional vector", while **v** represents the "positional vector". The equations above show the relation between vectors **ω** and **v** and the Plücker coordinates l_{ij}. In 1871 Klein [2] derived that $\boldsymbol{\omega}^T \mathbf{v} = 0$, i.e. in the Plücker coordinates: $l_{23} * l_{41} + l_{31} * l_{42} + l_{12} * l_{43} = 0$.
If q is a point on a line $\mathbf{q}(t) = \mathbf{q}_1 + \boldsymbol{\omega} t$ given by the Plücker coordinates, it must satisfy the equation $\boldsymbol{\omega} \times \mathbf{q} = \mathbf{v}$. A line given by two points in homogeneous coordinates is determined as: $\mathbf{q}(t) = \dfrac{\mathbf{v} \times \boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2} + \boldsymbol{\omega} t$.

Due to the **principle of duality** in E^3 we can exchange "point" and "plane" and therefore the same formula can be applied for an intersection computation of two planes as $\mathbf{L} = \boldsymbol{\rho}_1 \boldsymbol{\rho}_2^T - \boldsymbol{\rho}_2 \boldsymbol{\rho}_1^T$,. Now, the line $\boldsymbol{p}$ is given as $\mathbf{q}(t) = \dfrac{\mathbf{v} \times \boldsymbol{\omega}}{\|\boldsymbol{\omega}\|^2} + \boldsymbol{\omega} t$, where $\boldsymbol{\omega} = [l_{41}, l_{42}, l_{43}]^T$ $\mathbf{v} = [l_{23}, l_{31}, l_{12}]^T$.

Homogeneous coordinates can be used also for barycentric coordinates computation [5] and some additional hints can be found in [3], [6-8].

4. 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: $\lambda_i = -\xi_i / \xi_w, i = 1, \ldots, 3$, [Ska08a].

The area P of a triangle given by three points in E^2 can be easily computed as $\frac{1}{2} \boldsymbol{x}_1^T . (\boldsymbol{x}_2 \times \boldsymbol{x}_3) / (w_1 w_2 w_3)$.

$$P = \frac{1}{2} \boldsymbol{x}_1^T . (\boldsymbol{x}_2 \times \boldsymbol{x}_3) / (w_1 w_2 w_3) = det \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ w_1 & w_2 & w_3 \end{pmatrix} \Big/ (w_1 w_2 w_3)$$

Similarly a volume of a tetrahedron given by four points in E^3 can be computed as $\frac{1}{6} \boldsymbol{x}_1^T . (\boldsymbol{x}_2 \times \boldsymbol{x}_3 \times \boldsymbol{x}_4) / (w_1 w_2 w_3 w_4)$.

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.

$$G = (\boldsymbol{T}\boldsymbol{a}_1)^T(\boldsymbol{T}^{-1})^T(\boldsymbol{a}_2 \times \boldsymbol{a}_3)/det(\boldsymbol{T}) = \boldsymbol{a}_1^T\boldsymbol{T}^T(\boldsymbol{T}^{-1})^T(\boldsymbol{a}_2 \times \boldsymbol{a}_3)/det(\boldsymbol{T}) = \boldsymbol{a}_1^T(\boldsymbol{a}_2 \times \boldsymbol{a}_3)/det(\boldsymbol{T})$$

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.

5. IMPLEMENTATION ASPECTS

What is very important for today's applicability is the robustness and speed of computations. It means that also numerical methods should consider implementation aspects as well. The cross product is defined directly in Gg/HLSL on GPU. The cross product in 4D can be easily implemented in Cg/HLSL on 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));
    // or a.y=dot(x1.xzw, cross(x3.xzw, x2.xzw));
    a.z=dot(x1.xyw, cross(x2.xyw, x3.xyw));
    a.w=-dot(x1.xyz, cross(x2.xyz, x3.xyz));
    // or a.w=dot(x1.xyz, cross(x3.xyz, x2.xyz));

    return a;
}
```

or more compactly

Received