SIGGRAPH Asia 2012

Projective Geometry and Duality for Graphics, Games and Visualization

An Introductory Course



Singapore

Vaclav Skala

University of West Bohemia, Plzen, Czech Republic
VSB Technical University, Ostrava, Czech Republic
http://www.VaclavSkala.eu

SIGGRAPH Asia 2012

An overview

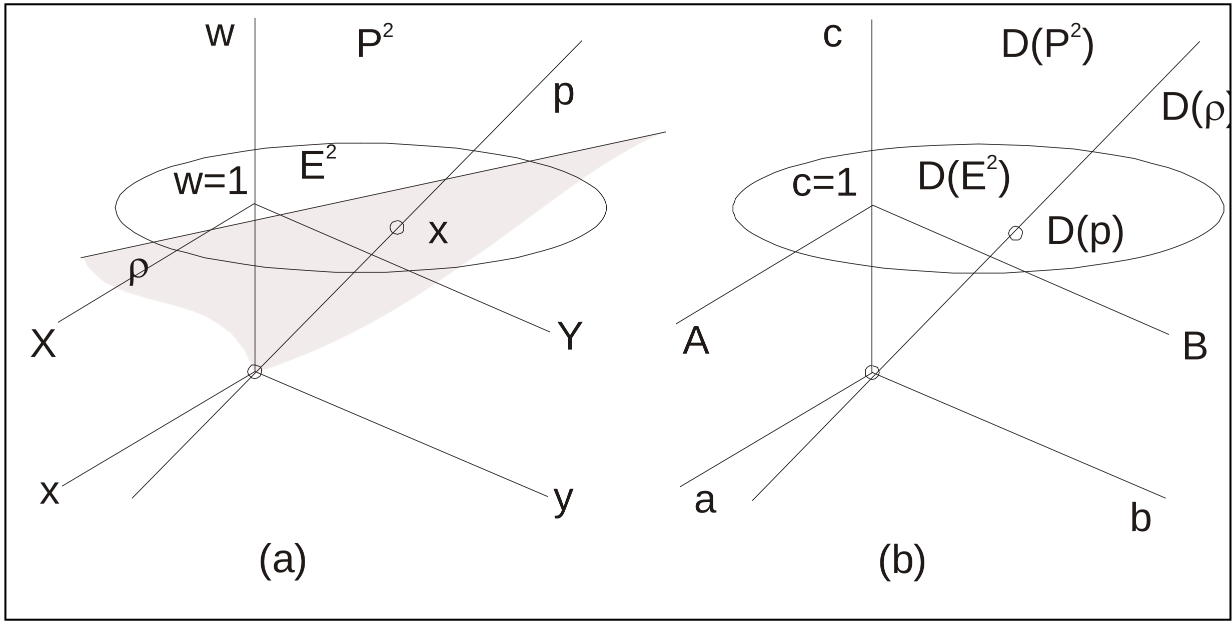
- Euclidean space and projective extension
- Principle of duality and its applications
- Geometric computation in the projective space
- Intersection of two planes in E3 with additional constrains
- Barycentric coordinates and intersections
- Interpolation and intersection algorithms
- Implementation aspects and GPU
- Conclusion and summary

SIGGRAPH Asia 2012

Projective Space

$\boldsymbol{X} = [X, Y]^T \quad \boldsymbol{X} \in E^2$

$\boldsymbol{x} = [x, y: w]^T \quad \boldsymbol{x} \in P^2$



Conversion:

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

$\& \; w \neq 0$

If $w = 0$ then $\boldsymbol{x}$ represents “an ideal point” - a point in infinity, i.e. it is a directional vector.

The Euclidean space E^2 is represented as a plane $w = 1$.

Points and vectors

- Vectors **are "freely movable"** – not having a fixed position

$$\boldsymbol{a}_1 = [x_1, y_1 : 0]^T$$

- Points are **not "freely movable"** – they are fixed to an origin of the current coordinate system

$$\boldsymbol{x}_1 = [x_1, y_1 : w_1]^T \quad \text{and} \quad \boldsymbol{x}_2 = [x_2, y_2 : w_2]^T$$

usually in textbooks $w_1 = w_2 = 1$

A vector $\boldsymbol{A} = \boldsymbol{X}_2 - \boldsymbol{X}_1$ in the Euclidean coordinate system - **CORRECT**

Horrible "construction" DO NOT USE IT – IT IS TOTALLY WRONG

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [x_2 - x_1, y_2 - y_1 : w_2 - w_1]^T = [x_2 - x_1, y_2 - y_1 : 1 - 1]^T$$
$$= [x_2 - x_1, y_2 - y_1 : 0]^T$$

This was presented as "How a vector" is constructed in the projective space P^k in a textbook!! **WRONG, WRONG, WRONG**

This construction was found in SW as well!!

SIGGRAPH Asia 2012

Points and vectors

A vector given by two points in the projective space

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

This is the **CORRECT SOLUTION**, but what is the interpretation?

A "difference" of coordinates of two points is a vector in the mathematical meaning and $w_1\, w_2$ is a "scaling" factor actually

In the projective representation (if the vector length matters)

$$\boldsymbol{a} = \boldsymbol{x}_2 - \boldsymbol{x}_1 = [w_1 x_2 - w_2 x_1, w_1 y_2 - w_2 y_1 : w_1\, w_2]^T$$
$$\triangleq \left[\frac{w_1 x_2 - w_2 x_1}{w_1\, w_2}, \frac{w_1 y_2 - w_2 y_1}{w_1\, w_2} : 0\right]^T$$

We have to strictly distinguish if we are working with points, i.e. vector as a data structure represents the coordinates, or with a vector in the mathematical meaning stored in a vector data structure.

VECTORS x FRAMES

Duality

For simplicity, let us consider a line p defined as:

$aX + bY + d = 0$

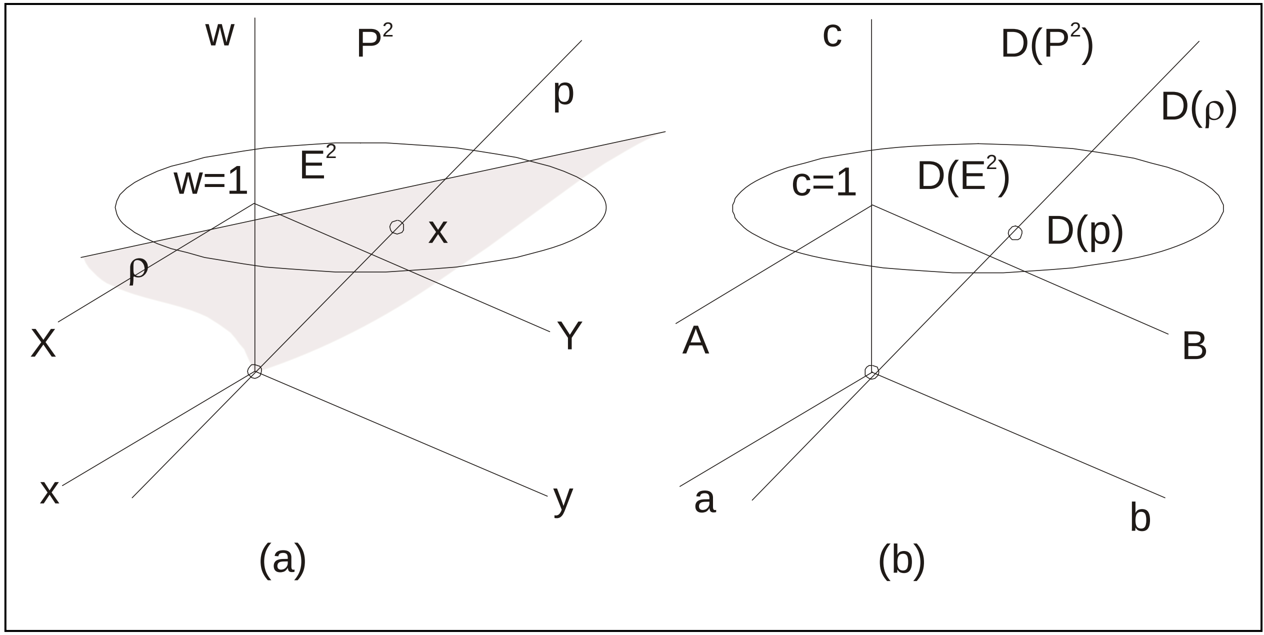
We can multiply it by $w \neq 0$ and we get:

$$ax + by + dw = 0$$

i.e. $\boldsymbol{p}^T\boldsymbol{x} = 0$

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

$\boldsymbol{x} = [\, x, y{:}\ w]^{\mathrm{T}} = [\, wX, wY{:}\ w]^{\mathrm{T}}$



A line $p \in E^2$ is actually a plane in the projective space P^2 (point $[0,0{:}0]^{\mathrm{T}}$ excluded)

Duality

From the mathematical notation $\boldsymbol{p}^T\boldsymbol{x} = 0$

we cannot distinguish whether ***p*** is a line and ***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.

This helps a lot to solve some geometrical problems.

Examples of dual objects and operators

	Primitive	Dual primitive
E^2	Point Line	Line Point
E^3	Point Plane	Plane Point

Operator	Dual operator
Join Intersect	Intersect Join

***Computational sequence for a problem is the same for a dual problem*.**

Definition 1

The cross product of the two vectors

$\mathbf{x}_1 = [x_1, y_1 : w_1]^T$ and $\mathbf{x}_2 = [x_2, y_2 : w_2]^T$

is defined as:

$$\boldsymbol{x}_1 \times \boldsymbol{x}_2 = det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}$$

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

Please, note that homogeneous coordinates are used.

Theorem 1

Let two points $\mathbf{x}_1$ and $\mathbf{x}_2$ be given in the projective space. Then the coefficients of the ***p*** line, which is defined by those two points, are determined as the cross product of their homogeneous coordinates

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 = [a, b : d]^T$$

Proof 1

Let the line $\boldsymbol{p} \in E^2$ be defined in homogeneous coordinates as (coefficient d is used intentionally to have the same symbol representing a “distance” of the element from the origin for lines and planes)

$$ax + by + dw = 0$$

We are actually looking for a solution to the following equations:

$$\boldsymbol{p}^T \boldsymbol{x}_1 = 0 \qquad\qquad \boldsymbol{p}^T \boldsymbol{x}_2 = 0$$

where: $\mathbf{p} = [a,b : d]^T$

SIGGRAPH Asia 2012

It means that any point $\boldsymbol{x}$ that lies on the *p* line must satisfy both the equation above and the equation $\mathbf{p}^T\mathbf{x} = 0$ in other words the $\boldsymbol{p}$ vector is defined as

$$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 = det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}$$

We can write

$$(\boldsymbol{x}_1 \times \boldsymbol{x}_2)^{\boldsymbol{T}} \boldsymbol{x} = \boldsymbol{0} \qquad \text{i.e.} \quad det \begin{bmatrix} a & b & c \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0$$

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

we get the line coefficients of the line *p* as:

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

Note: for w = 1 we get the standard cross product formula and the cross product defines the *p* line, i.e. $\mathbf{p} = \mathbf{x}_1 \times \mathbf{x}_2$
where: $\mathbf{p} = [a,b : d]^T$

- An intersection of two lines => ***A x*** = ***b***
- A line given by two points => ***A x*** = ***0***

Those problems are DUAL,
why algorithms are different??

Cross product is equivalent to a solution of a linear system of equations !

DUALITY APPLICATION

In the projective space P^2 points and lines are dual. Due to duality we can directly intersection of two lines as

$$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2 = det \begin{bmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{bmatrix} = [x, y: w]^T$$

If the lines are parallel or close to parallel, the homogeneous coordinate $w \rightarrow 0$ and a user has to make a decision – so there is sequence in the code like $\boldsymbol{if}\ abs(\mathrm{det}(..)) \leq eps\ \boldsymbol{then} \ldots$
Generally computation can continue even if $w \rightarrow 0$ if projective space is used.

DISTANCE

Geometry is strongly connected with distances and their measurement, geometry education is strictly sticked to the Euclidean geometry, where the distance is measured as

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} \quad \text{, resp.} \quad d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2} \; .$$

This concept is convenient for a solution of basic geometric problems, but in many cases it results into quite complicated formula and there is a severe question of stability and robustness in many cases.

The main objection against the projective representation is that there is no metric.

The distance of two points can be easily computed as

$$dist = \sqrt{\xi^2 + \eta^2}/(w_1 w_2)$$

where: $\xi = w_1 x_2 - w_2 x_1 \; \eta = w_1 y_2 - w_2 y_1$

Also a distance of a point $\boldsymbol{x}_0$ from a line in E^2 can be computed as

$$dist = \frac{\boldsymbol{a}^T \boldsymbol{x}_0}{w_0 \sqrt{a^2 + b^2}}$$

where: $\boldsymbol{x_0} = [x_0, y_0 : w_0]^T \qquad \boldsymbol{a} = [a, b : d]^T$

The extension to E^3/P^3 is simple and the distance of a point $\boldsymbol{x}_0$ from a plane in E^3 can be computed as

$$dist = \frac{\boldsymbol{a}^T \boldsymbol{x}_0}{w_0 \sqrt{a^2 + b^2 + c^2}}$$

where: $\boldsymbol{x_0} = [x_0, y_0, z_0 : w_0]^T \qquad \boldsymbol{a} = [a, b, c : d]^T$.

SIGGRAPH Asia 2012

In many cases we do not need actually a *distance*, e.g. for a decision which object is closer, and *distance*^{2} can be used instead, i.e. for the E^2 case

$$dist^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2(a^2 + b^2)} = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2\, \boldsymbol{n}^T \boldsymbol{n}}$$

where: $\boldsymbol{a} = [a, b : d]^T = [\boldsymbol{n} : d]^T$ and the normal vector $\boldsymbol{n}$ is not normalized

If we are comparing distances from the same line *p* we can use "*pseudo-distance*" for comparisons

$$(pseudo - dist)^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2}$$

Similarly in the case of E^3

$$dist^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2(a^2 + b^2 + c^2)} = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2\, \boldsymbol{n}^T \boldsymbol{n}} \quad \text{and} \quad (pseudo - dist)^2 = \frac{(\boldsymbol{a}^T \boldsymbol{x}_0)^2}{w_0{}^2}$$

where: $\boldsymbol{a} = [a, b, c : d]^T = [\boldsymbol{n} : d]^T$

SIGGRAPH Asia 2012

Computation in Projective Space

- Cross product definition
- A plane $\boldsymbol{\rho}$ is determined as a cross product of three given points

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

Due to the duality

- An intersection point $\boldsymbol{x}$ of three planes is determined as a cross product of three given planes

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

Computation of generalized cross product is equivalent to a solution of a linear system of equations => no division operation!

SIGGRAPH Asia 2012

Geometric transformations with points

(note $X = {}^{x}/_{w}$, $Y = {}^{y}/_{w}$, $w \neq 0$):

Translation by a vector $(A, B) \triangleq [a, b: c]^T$, i.e. $A = a/c$, $B = b/c$, $c \neq 0$:

In the Euclidean space: $\boldsymbol{x}' = \boldsymbol{T}\boldsymbol{x}$

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & A \\ 0 & 1 & B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} x + Aw \\ y + Bw \\ w \end{bmatrix} \triangleq \begin{bmatrix} x/w + A \\ y/w + B \\ 1 \end{bmatrix} = \begin{bmatrix} X + A \\ Y + A \\ 1 \end{bmatrix}$$

In the projective space: $\boldsymbol{x}' = \boldsymbol{T}'\boldsymbol{x}$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} c & 0 & a \\ 0 & c & b \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} cx + aw \\ cy + bw \\ cw \end{bmatrix} \triangleq \begin{bmatrix} (cx + aw)/(cw) \\ (cy + bw)/(cw) \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{x}/_{w} + {}^{a}/_{c} \\ {}^{y}/_{w} + {}^{b}/_{c} \\ 1 \end{bmatrix} = \begin{bmatrix} X + A \\ Y + B \\ 1 \end{bmatrix}$$

and $\det(\boldsymbol{T}') = c^3$. For $c = 1$ we get a standard formula

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & A \\ 0 & 1 & B \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation by an angle $(cos\varphi, sin\varphi) = \left(\frac{a}{c}, \frac{b}{c}\right) \triangleq [a, b: c]^T$

In the Euclidean space: $\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x}$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \triangleq \begin{bmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

In the projective space: $\boldsymbol{x}' = \boldsymbol{R}'\boldsymbol{x}$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & -b & 0 \\ b & a & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} ax - by \\ bx + ay \\ cw \end{bmatrix} \triangleq$$

$$\begin{bmatrix} (ax - by)/(cw) \\ (bx + ay)/(cw) \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x}{w}\frac{a}{c} - \frac{y}{w}\frac{b}{c} \\ \frac{x}{w}\frac{b}{c} + \frac{y}{w}\frac{a}{c} \\ 1 \end{bmatrix} = \begin{bmatrix} Xcos\varphi - Ysin\varphi \\ Xsin\varphi + Ycos\varphi \\ 1 \end{bmatrix}$$

as $c^2 = (a^2 + b^2)$ by definition, $\det(\boldsymbol{R}') = (a^2 + b^2)c = c^3$

SIGGRAPH Asia 2012

Scaling by a factor $(S_x, S_y) = (\frac{s_x}{w_s}, \frac{s_y}{w_s}) \triangleq [s_x, s_y : w_s]^T$

$$\boldsymbol{x}' = \boldsymbol{S}\boldsymbol{x} \qquad \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\boldsymbol{x}' = \boldsymbol{S}'\boldsymbol{x} \qquad \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & w_s \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\det(\boldsymbol{S}') = s_x \, s_y \, w_s$$

It is necessary to note that the determinant of a transformation matrix $\boldsymbol{Q}$, i.e. matrices $\boldsymbol{T}', \boldsymbol{R}', \boldsymbol{S}'$, is $\det(\boldsymbol{Q}) \neq 1$ in general, but as the formulation is in the projective space, there is no need to “normalize” transformations to $\det(\boldsymbol{Q}) = 1$ even for rotation.

It can be seen that if the parameters of a geometric transformation are given in the homogeneous coordinates, no division operation is needed at all.

Transformation of lines and planes

	E^2	E^3
	$\boldsymbol{p} = \boldsymbol{x}_1 \times \boldsymbol{x}_2$	$\boldsymbol{\rho} = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \times \boldsymbol{x}_3$
Dual problem	$\boldsymbol{x} = \boldsymbol{p}_1 \times \boldsymbol{p}_2$	$\boldsymbol{x} = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_3$

In graphical applications position of points are changed by an interaction, i.e. $\boldsymbol{x}' = \boldsymbol{T}\boldsymbol{x}$.

The question is how coefficients of a line, resp. a plane are changed without need to recompute them from the definition.

It can be proved that

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

or

$$\boldsymbol{\rho}' = (\boldsymbol{T}\boldsymbol{x}_1) \times (\boldsymbol{T}\boldsymbol{x}_2) \times (\boldsymbol{T}\boldsymbol{x}_3) = det(\boldsymbol{T})(\boldsymbol{T}^{-1})^T \boldsymbol{\rho}$$

Transformation of lines and planes

As the computation is made **in the projective space** we can write

$$\boldsymbol{p}' = (\boldsymbol{T}^{-1})^T \boldsymbol{p} = [a',\ b' : d']^T$$ for lines in E^2

or

$$\boldsymbol{\rho}' = (\boldsymbol{T}^{-1})^T \boldsymbol{\rho} = [a',\ b', c' : d']^T$$ for planes in E^3

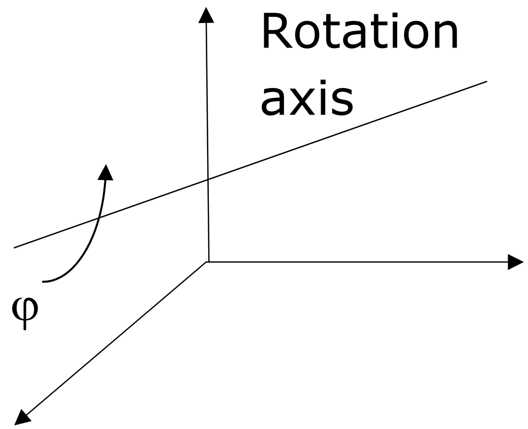
THIS IS SIMPLIFICATION OF COMPUTATIONS

Transformation matrices for lines, resp. for planes are **DIFFERENT** from transformations for points!

SIGGRAPH Asia 2012

Transformation of lines and planes

Transformation about a general axis in E^3 / P^3



Usually used transformation (**T** is translation):

$$\boldsymbol{Q} = \boldsymbol{T}^{-1}\boldsymbol{R}_{zx}^{-1}\boldsymbol{R}_{yz}^{-1}\boldsymbol{R}(\varphi)\boldsymbol{R}_{zx}\boldsymbol{R}_{xy}\boldsymbol{T}$$

$$\boldsymbol{R}_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{\sqrt{b^2+c^2}} & \frac{-b}{\sqrt{b^2+c^2}} & 0 \\ 0 & \frac{b}{\sqrt{b^2+c^2}} & \frac{c}{\sqrt{b^2+c^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

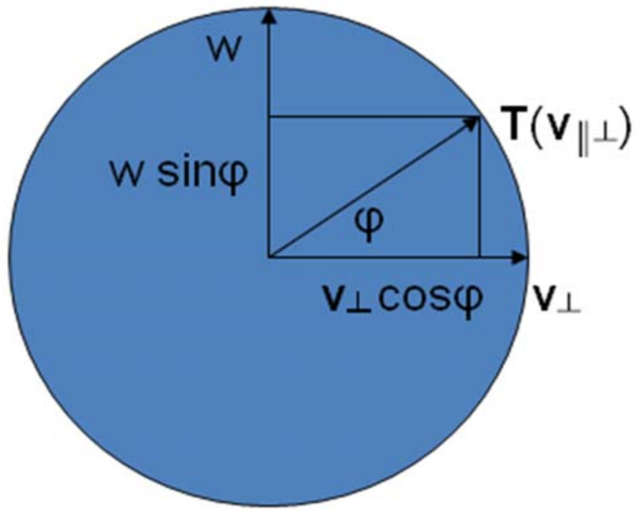
$\boldsymbol{a} = [a, b, c]^T$ is an axis directional vector

This is unstable if $\sqrt{b^2 + c^2} \rightarrow 0$ and not precise if $b^2 \gg c^2$ or vice versa

That is generally computationally complex and unstable as
a user has to select which axis is to be used for a rotation

Transformation of lines and planes

Transformation about an axis $\boldsymbol{n}$
in the Euclidean space E^3



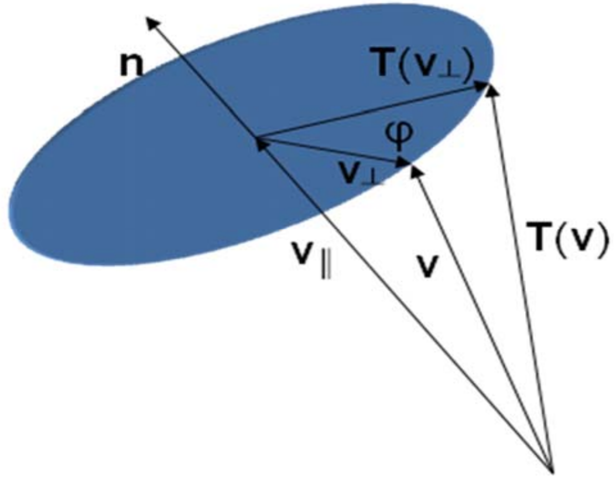
$$\boldsymbol{X} = \boldsymbol{X}\,cos\varphi + (1 - cos\varphi)(\,\boldsymbol{n}^T\boldsymbol{X}).\boldsymbol{n} + (\boldsymbol{n} \times \boldsymbol{X})sin\varphi$$

$$\boldsymbol{Q} = \boldsymbol{I}cos\varphi + (1 - cos\varphi)(\boldsymbol{n}\otimes\boldsymbol{n}) + \boldsymbol{W}sin\varphi$$

where: $\boldsymbol{n}\otimes\boldsymbol{n} = \boldsymbol{n}\,.\boldsymbol{n}^T$ is a matrix.
In the Euclidean space E^3 the vector $\boldsymbol{n}$ has to be normalized

The matrix $\boldsymbol{W}$ is defined as: $\boldsymbol{W}\boldsymbol{v} = \boldsymbol{w} \times \boldsymbol{v}$

$$\boldsymbol{W} = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$



Computation in Projective Space

Interpolation

Linear parametrization

$$\mathbf{X}(t) = \mathbf{X}_0 + (\mathbf{X}_1 - \mathbf{X}_0)t \quad t \in (-\infty, \infty)$$

Non-linear (monotonous) parametrization

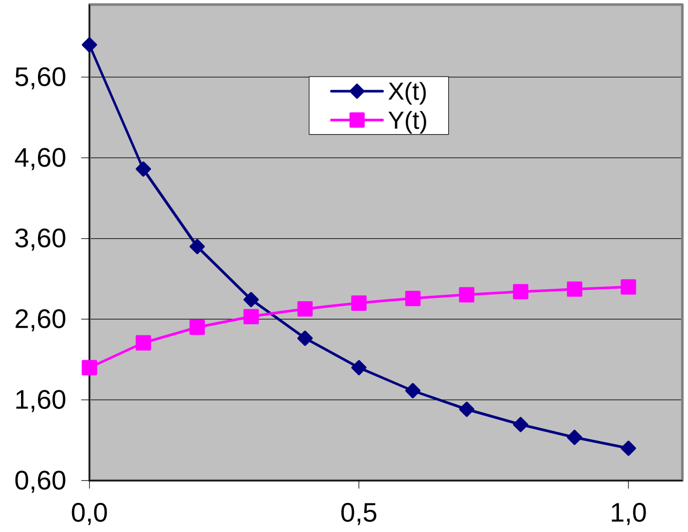
$$\mathbf{x}(t) = \mathbf{x}_0 + (\mathbf{x}_1 - \mathbf{x}_0)t \quad t \in (-\infty, \infty)$$

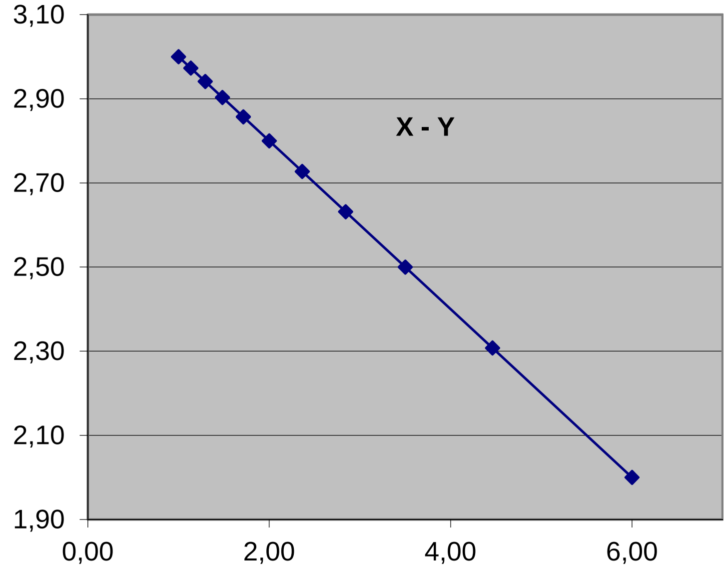
$$x(t) = x_0 + (x_1 - x_0)t \qquad y(t) = y_0 + (y_1 - y_0)t$$

$$z(t) = z_0 + (z_1 - z_0)t \qquad w(t) = w_0 + (w_1 - w_0)t$$

- t means that we can interpolate using homogeneous coordinates without a need of “normalization” to E^k !!
- Homogeneous coordinate $w \geq 0$

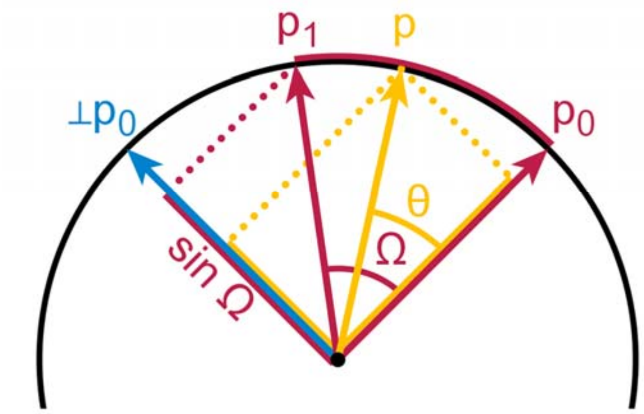
In many algorithms, we need “monotonous” parameterization, only





SIGGRAPH Asia 2012

Computation in Projective Space



Spherical interpolation

$$slerp(\boldsymbol{x}_0, \boldsymbol{x}_1, t) = \frac{\sin[(1-t)\Omega]}{\sin\Omega}\boldsymbol{x}_0 + \frac{\sin[t\Omega]}{\sin\Omega}\boldsymbol{x}_1$$

Instability occurs if $\Omega \rightarrow k\pi$.

Mathematically formula is correct,

in practice the **code is generally incorrect**! $[\ \frac{0}{0}\ ]$

$$slerp_p(\boldsymbol{x}_0, \boldsymbol{x}_1, t) = \left[\frac{\sin[(1-t)\Omega]\boldsymbol{x}_0 + \sin[t\Omega]\boldsymbol{x}_1}{\sin\Omega}\right]$$
$$= [\sin[(1-t)\Omega]\boldsymbol{x}_0 + \sin[t\Omega]\boldsymbol{x}_1 : \sin\Omega]^T$$

Homogeneous coordinates
=> better numerical stability

Homogeneous
coordinate

Computation in Projective Space

Intersection line – plane

$$\mathbf{x}(t) = \mathbf{x}_0 + (\mathbf{x}_1 - \mathbf{x}_0)t \qquad t \in (-\infty, \infty)$$

$$\mathbf{a}^T \mathbf{x} = 0 \qquad ax + by + cz + d = 0$$

$$\mathbf{a} = [a, b, c, d]^T \qquad \mathbf{S} = \mathbf{X}_1 - \mathbf{X}_0$$

$$t = -\frac{\mathbf{a}^T \mathbf{x}_0}{\mathbf{a}^T \mathbf{s}}$$

$$\tau = -\mathbf{a}^T \mathbf{x}_0 \qquad \tau_w = \mathbf{a}^T \mathbf{s}$$

$$\mathbf{t} = [\tau : \tau_w] \quad \textbf{if } \tau_w \leq 0 \textbf{ then } \mathbf{t} := -\mathbf{t}$$

TEST

$$\textbf{if } t > t_{\min} \textbf{then}.....$$

$$\textbf{if } \tau * \tau_{\min_w} > \tau_w * \tau_{\min} \textbf{ then}..... \text{ condition } \tau \geq 0$$

An intersection of a plane with a line in E^2 / E^3 can be computed efficiently

Comparison operations must be modified !!!

Cyrus-Beck line clipping algorithm 10-25% faster

SIGGRAPH Asia 2012

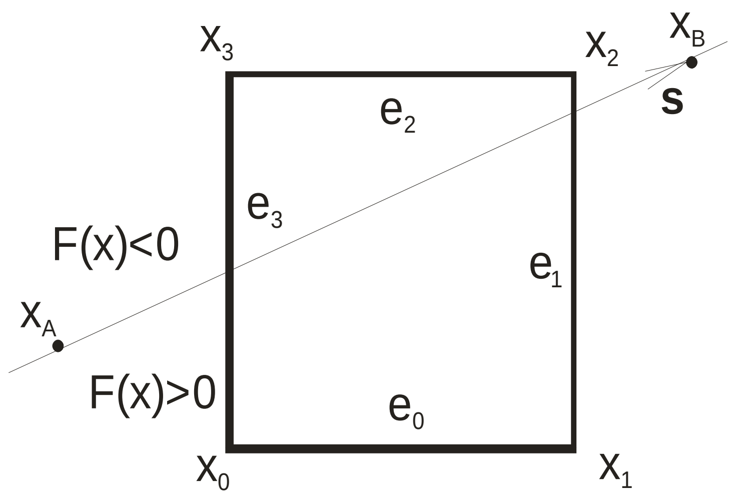
Line Clipping

procedure CLIP_L;
{ $\boldsymbol{x}_A$, $\boldsymbol{x}_B$ – in homogeneous coordinates }
{ The **EXIT** ends the procedure }
{ **input**: $\boldsymbol{x}_A$, $\boldsymbol{x}_B$; $\boldsymbol{x}_A=[x_A,y_A,1]^T$ $\boldsymbol{p} = [a,b,c]^T$}
begin
{1}$\boldsymbol{p} := \boldsymbol{x}_A \times \boldsymbol{x}_B$; { ax+by+c = 0}
{2}**for** k:=0 **to** N-1 **do** { $\boldsymbol{x}_k=[x_k,y_k,1]^T$ }
{3} **if** $\boldsymbol{p}^T\boldsymbol{x}_k \geq 0$ **then** $\mathbf{c}_k$:=1 **else** c_k:=0;
{4}**if** **c** = $[0000]^T$ **or** **c** = $[1111]^T$ **then** **EXIT**;
{5}i:= TAB1[**c**]; j:= TAB2[**c**];
{6}$\boldsymbol{x}_A := \boldsymbol{p} \times \boldsymbol{e}_i$; $\boldsymbol{x}_B := \boldsymbol{p} \times \boldsymbol{e}_j$;
{7}**DRAW** ($\boldsymbol{x}_A$; $\boldsymbol{x}_B$) {$\boldsymbol{e}_i$ – *i-th edge* }
end {CLIP_L}

SIMPLE, ROBUST and FAST

Line clipping algorithms in E^2

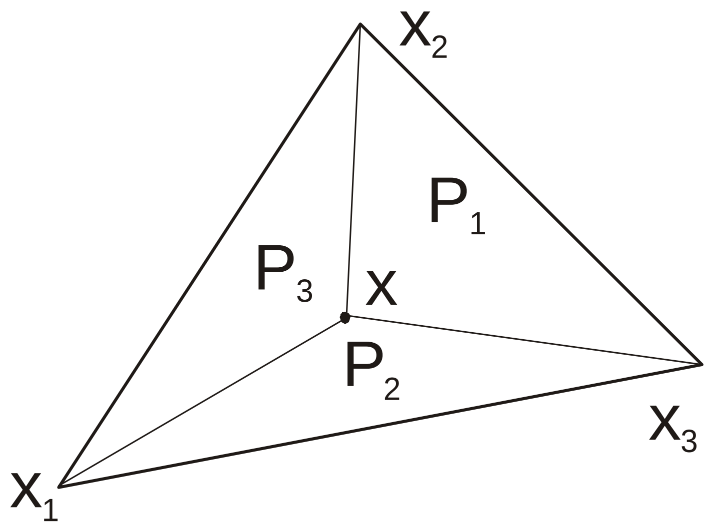
- Cohen-Sutherland
- Liang-Barsky
- Hodgman
- Skala – modification of Clip_L for line segments



Computation in Projective Space - Barycentric coordinates

Let us consider a triangle with vertices $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$,

A position of any point $\mathbf{X} \in E^2$ can be expressed as



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

$$a_1 Y_1 + a_2 Y_2 + a_3 Y_3 = Y$$

additional condition

$$a_1 + a_2 + a_3 = 1 \quad 0 \le a_i \le 1$$

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

Linear system must be solved

If points $\mathbf{x}_i$ are given as $[x_i, y_i, z_i: w_i]^T$ and $w_i \neq 1$then $\mathbf{x}_i$ must be "normalized" to $w_i = 1$, i.e. 4 * 3 = 12 division operations

Computation in Projective Space

$$b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X = 0$$

$$b_1 Y_1 + b_2 Y_2 + b_3 Y_3 + b_4 Y = 0$$

$$b_1 + b_2 + b_3 + b_4 = 0$$

$$b_i = -a_i b_4 \quad i = 1,\dots,3 \qquad b_4 \neq 0$$

Rewriting

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

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

$$\mathbf{b} = \left[b_1, b_2, b_3, b_4 \right]^T$$

$$\boldsymbol{\xi} = \left[X_1, X_2, X_3, X \right]^T$$

$$\boldsymbol{\eta} = \left[Y_1, Y_2, Y_3, Y \right]^T$$

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

Solution of the linear system of equations (LSE) is equivalent to generalized cross product

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

Computation in Projective Space

if $w_i \neq 1$ or $w_i = 1$

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

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

$$\mathbf{b} = \left[b_1, b_2, b_3, b_4 \right]^T$$

$$\boldsymbol{\xi} = \left[x_1, x_2, x_3, x \right]^T$$

$$\boldsymbol{\eta} = \left[y_1, y_2, y_3, y \right]^T$$

$$\mathbf{w} = \left[w_1, w_2, w_3, w \right]^T$$

$$0 \leq (-b_1 : w_2 w_3 w) \leq 1$$

$$0 \leq (-b_2 : w_3 w_1 w) \leq 1$$

$$0 \leq (-b_3 : w_1 w_2 w) \leq 1$$

=> **new entities**:

projective scalar, projective vector

(Skala,V.: Barycentric coordinates computation in homogeneous coordinates, Computers&Graphics, 2008)

Computation in Projective Space

Line in E3 as Two Plane Intersection

Standard formula in the Euclidean space

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

Line given as an intersection of two planes

$$\boldsymbol{s} = \boldsymbol{n}_1 \times \boldsymbol{n}_2 \qquad \boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{x}_0 + \boldsymbol{s}t$$

$$x_0 = \frac{d_2 \begin{vmatrix} b_1 & c_1 \\ b_3 & c_3 \end{vmatrix} - d_1 \begin{vmatrix} b_2 & c_2 \\ b_3 & c_3 \end{vmatrix}}{DET} \qquad y_0 = \frac{d_2 \begin{vmatrix} a_3 & c_3 \\ a_1 & c_1 \end{vmatrix} - d_1 \begin{vmatrix} a_3 & c_3 \\ a_2 & c_2 \end{vmatrix}}{DET}$$

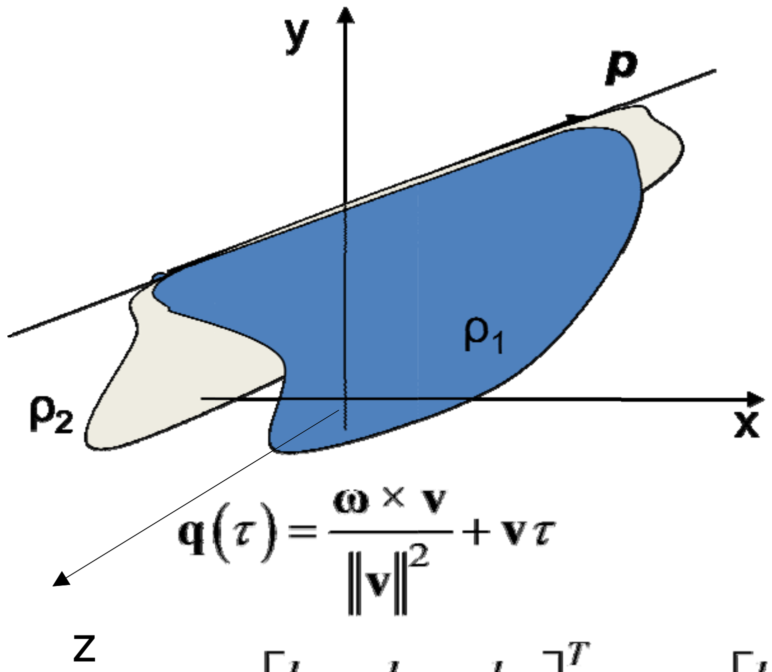
$$z_0 = \frac{d_2 \begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix} - d_1 \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}}{DET} \qquad DET = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$$

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.

Computation in Projective Space

Line in E3 as Two Plane Intersection

- Standard formula in the Euclidean space
- Plücker's coordinates – a line is given by two points. Due to the DUALITY - point is dual to a plane and vice versa – the intersection of two planes can be computed as a dual problem – but computationally **expensive** computation



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

$$\boldsymbol{\omega} = \begin{bmatrix} l_{41} & l_{42} & l_{43} \end{bmatrix}^T \quad \mathbf{v} = \begin{bmatrix} l_{23} & l_{31} & l_{12} \end{bmatrix}^T$$

$$\mathbf{L} = \mathbf{a}_0\mathbf{a}_1^T - \mathbf{a}_1\mathbf{a}_0^T \quad \textit{tensor product} - \textit{matrix}$$

$$\mathbf{a}_i = \left[a_i, b_i, c_i, d_i\right]^T \quad i = 1,2$$

- Projective formulation and simple computation

Computation in Projective Space

Line in E3 as Two Plane Intersection

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

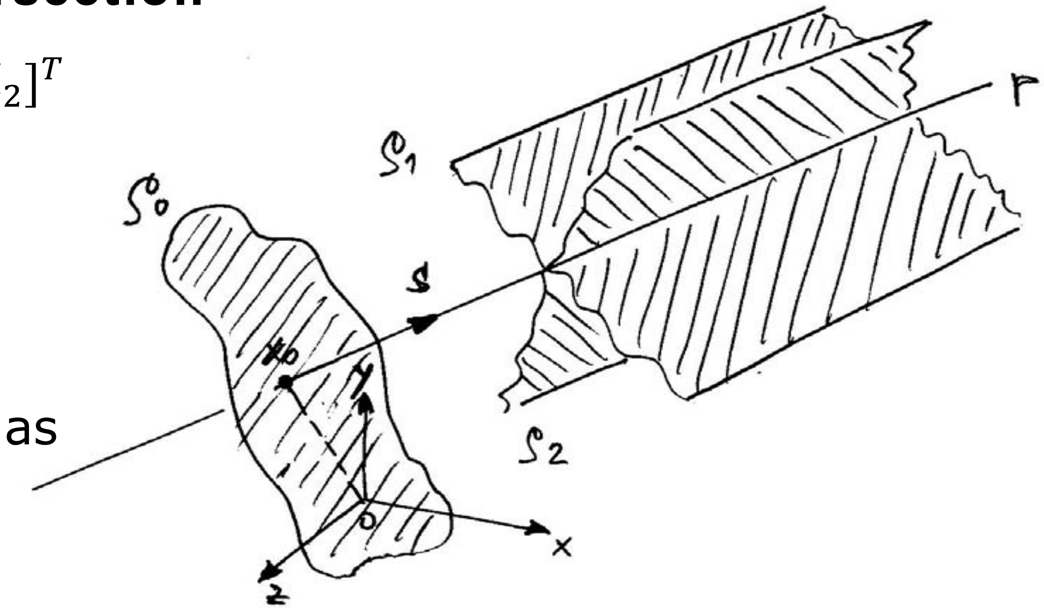
normal vectors are

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

directional vector of a line
of two planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_1$ is given as

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

"starting" point $\boldsymbol{x}_0$ **???**



A plane $\boldsymbol{\rho}_0$ passing the origin with a normal vector $\boldsymbol{s}$, $\boldsymbol{\rho}_0 = [a_0, b_0, c_0 : 0]^T$

The point $\boldsymbol{x}_0$ is defined as $\boldsymbol{x}_0 = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_0$

How simple formula supporting matrix-vector architectures like GPU and parallel processing

Compare the standard and projective formulas

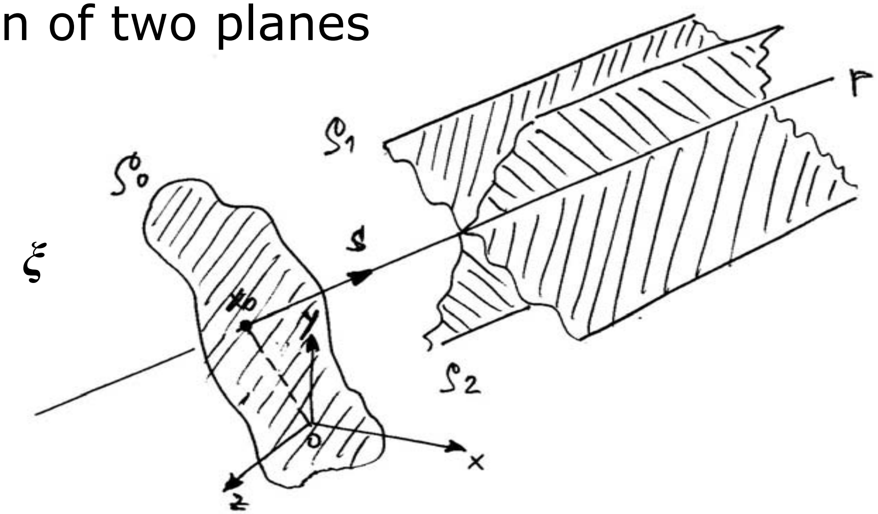
Computation in Projective Space – the nearest point

Find the nearest point on an intersection of two planes to the given point ξ

Simple solution:

Translate planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ so the ξ is in the origin

- Compute intersection of two planes i.s. $\boldsymbol{x}_0$ and $\boldsymbol{s}$
- Translate $\boldsymbol{x}_0$ using $\boldsymbol{T}^{-1}$



Known solution using Lagrange multipliers

$$\begin{bmatrix} 2 & 0 & 0 & n_{1x} & n_{2x} \\ 0 & 2 & 0 & n_{1y} & n_{2y} \\ 0 & 0 & 2 & n_{1z} & n_{2z} \\ n_{1x} & n_{1y} & n_{1z} & 0 & 0 \\ n_{2x} & n_{2y} & n_{2z} & 0 & 0 \end{bmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} 2p_{0x} \\ 2p_{0y} \\ 2p_{0z} \\ \vec{p}_1 \cdot \vec{n}_1 \\ \vec{p}_2 \cdot \vec{n}_2 \end{pmatrix}$$

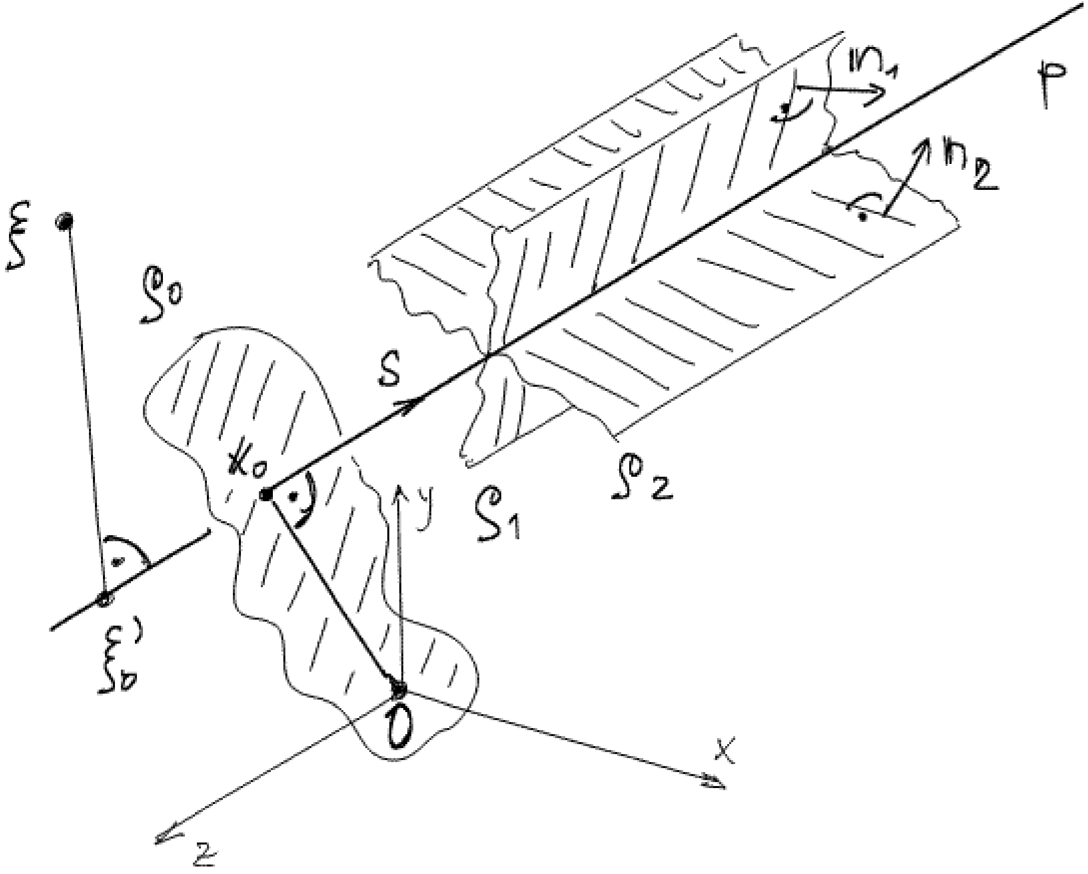
Again – an elegant solution, simple formula supporting matrix-vector architectures like GPU and parallel processing

Solution DETAILS next

Krumm,J.: Intersection of Two Planes, Microsoft Research

The closest point to an intersection of two planes

In some applications we need to find a closest point on a line given as an intersection of two planes. We want to find a point $\boldsymbol{\xi}_0{}'$, the closest point to the given point $\boldsymbol{\xi}$, which lies on an intersection of two planes



$\boldsymbol{\rho}_1 \triangleq [\boldsymbol{n}_1^T\colon d_1]^T$ and $\boldsymbol{\rho}_2 \triangleq [\boldsymbol{n}_2^T\colon d_2]^T$

This problem was recently solved by using Lagrange multipliers and an optimization approach.

This leads to a solution of a system of linear equations with 5 equations.

Solution in the projective space

1. Translates the given point $\boldsymbol{\xi} = [\xi_x, \xi_y, \xi_z : 1]^T$ to the origin – matrix $\boldsymbol{Q}$
2. Compute parameters of the given planes $\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2$ after the transformation as $\boldsymbol{\rho}_1' = \boldsymbol{Q}^{-T}\boldsymbol{\rho}_1$ and $\boldsymbol{\rho}_2' = \boldsymbol{Q}^{-T}\boldsymbol{\rho}_2$,
3. Compute the intersection of those two planes $\boldsymbol{\rho}_1'$ and $\boldsymbol{\rho}_2'$
4. Transform the point $\boldsymbol{\xi}_0$ to the original coordinate system using transformation

$$\boldsymbol{n}_0 = \boldsymbol{n}_1 \times \boldsymbol{n}_2 \qquad \boldsymbol{\rho}_0 \triangleq [\boldsymbol{n}_0^T : 0]^T \qquad \boldsymbol{\xi}_0 = \boldsymbol{\rho}_1 \times \boldsymbol{\rho}_2 \times \boldsymbol{\rho}_0 \qquad \boldsymbol{\xi}_0' = \boldsymbol{Q}^{-1}\,\boldsymbol{\xi}_0$$

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

It is simple, easy to implement on GPU.

Computation in Projective Space

Disadvantages

- Careful handling with formula as the projective space
- "Oriented" projective space is to be used, i.e. $w \geq 0$; HW could support it
- Exponents of the homogeneous vector can overflow
  - o exponents should be normalized; HW could support it
  - o unfortunately not supported by the current hardware
  - o P_Lib – library for computation in the projective space - uses SW solution for normalization on GPU (C# and C++)

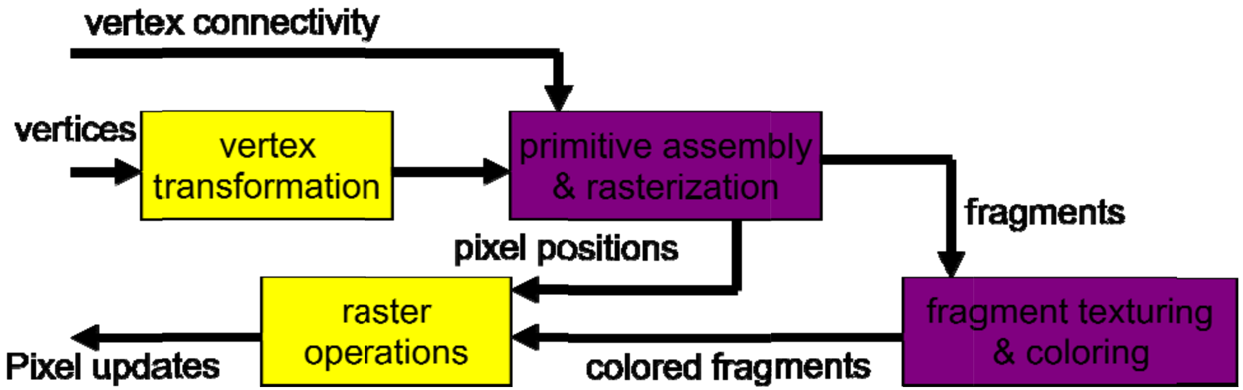
Computation in Projective Space

Advantages

- "Infinity" is represented
- No division operation is needed, a division operation can be hidden to the homogeneous coordinate
- Many mathematical formula are simpler and elegant
- One code sequence solve primary and dual problems
- Supports matrix – vector operations in hardware – like GPU etc.
- Numerical computation can be faster
- More robust and stable solutions can be achieved
- System of linear equations can be solved directly without division operation if exponent normalization is provided

Implementation aspects and GPU

- GPU (Graphical Processing Unit) -optimized for matrix-vector, vector-vector operation – especially for [x,y,z :w]
- Native arithmetic operations with homogeneous coordinates – without exponent "normalization"
- Programmable HW – parallel processing



Implementation aspects and GPU

4D cross product can be implemented in Cg/HLSL on GPU (not optimal implementation) as:

```
float4 cross_4D(float4 x1, float4 x2, float4 x3)
{ float4 a; # simple formula #
  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;
}
# more compact formula available #
```

Geometry algebra

$$\boldsymbol{ab} = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b} \qquad \text{in } E^3 \qquad \boldsymbol{ab} = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{b}$$

It is strange – result of a dot product is a scalar value while result of the outer product (cross product) is a vector.

What is $\boldsymbol{ab}$**???**

Please, for details see

- http://geometricalgebra.zcu.cz/
- GraVisMa - workshops on Computer Graphics, Computer Vision & Mathematics http://www.GraVisMa.eu
- WSCG – Conference on Computer Graphics, Computer Vision & Visualization http://www.wscg.eu

SIGGRAPH Asia 2012

Summary and conclusion

We have got within this course an understanding of:

- projective representation use for geometric transformations with points, lines and planes
- principle of duality and typical examples of dual problems, influence to computational complexity
- intersection computation of two planes in E3, dual Plücker coordinates and simple projective solution
- geometric problems solution with additional constrains
- intersection computations and interpolation algorithms directly in the projective space
- barycentric coordinates computation on GPU
- avoiding or postponing division operations in computations

Projective space representation supports matrix-vector architectures like GPU – faster, robust and easy to implement algorithms achieved

SIGGRAPH Asia 2012

References

- Skala,V.: Barycentric Coordinates Computation in Homogeneous Coordinates, Computers & Graphics, Elsevier, ISSN 0097-8493, Vol. 32, No.1, pp.120-127, 2008
- Skala,V.: Intersection Computation in Projective Space using Homogeneous Coordinates, Int. Journal of Image and Graphics, ISSN 0219-4678, Vol.7, No.4, pp.615-628, 2008
- Skala,V.: Length, Area and Volume Computation in Homogeneous Coordinates, Int. Journal of Image and Graphics, Vol.6., No.4, pp.625-639, ISSN 0219-4678, 2006
- Skala,V., Kaiser,J., Ondracka,V.: Library for Computation in the Projective Space, 6th Int.Conf. Aplimat, Bratislava, ISBN 978-969562-4-1, pp. 125-130, 2007
- Skala,V.: GPU Computation in Projective Space Using Homogeneous Coordinates , Game Programming GEMS 6 (Ed.Dickheiser,M.), pp.137-147, ISBN 1-58450-450-1, Charles River Media, 2006
- Skala,V.: A new approach to line and line segment clipping in homogeneous coordinates, The Visual Computer, ISSN 0178-2789, Vol.21, No.11, pp.905-914, Springer Verlag, 2005

- Generally Publications with on-line DRAFTs via http://www.VaclavSkala.eu

References related

- Yamaguchi,F.:Computer-Aided Geometric Design: A Totally Four-Dimensional Approach, Springer, 2002

- Agoston,M.K.: Computer Graphics and Geometric Modeling - Mathematics, ISBN 1-58233-817-2, Springer, 2005

SIGGRAPH Asia 2012

Additional references for Geometric Algebra and Conformal Geometry

- Bayro-Corrochano,E: Geometric Computing: For Wavelet Transforms, Robot Vision, Learning, Control and Action, Springer, 2010
- Bayro-Corrochano,E, Sobczyk,G.: Geometric Algebra with Application in Science and Engineering, Birkhauser, 2001
- Browne,J.: Grassmann Algebra - Draft Exploring Application of Extended Vector Algebra with Mathematica, Swinburn University of Technology, Melbourne, Australia, 2001
- Chevalley,C.: The Algebraic Theory of Spinors and Clifford Algebras, Springer, 1995
- Doran,Ch., Lasenby,A.: Geometric Algebra for Physicists, Cambridge Univ.Press, 2003
- Dorst,L., Fontine,D., Mann,S.: Geometric Algebra for Computer Science, Morgan Kaufmann, 2007
- Gu,X.D., Yau,S.-T.: Computational Conformal Geometry, International Press, Somerville, USA, 2008
- Hildenbrand,D.: Foundations of Geometric Algebra Computing, Springer 2012
- Li,H., Olver,P.J., Sommer,G.: Computer Algebra and Geometric Algebra with Applications, Springer
- Perwass,Ch.: Geometric Algebra with Applications in Engineering, Springer Verlag, 2008
- Sommer, G.: Geometric Computing with Clifford Algebras. Theoretical Foundations and Applications in Computer Vision and Robotics, Springer, Berlin, 2001
- Suter,J.: Geometric Algebra Primer [PDF] University of Twente, The Netherlands

SIGGRAPH Asia 2012

- Vince,J.: Geometric Algebra for Computer Science, Springer, 2008
- Vince,J.: Geometric Algebra: An Algebraic System for Computer Games and Animation, Springer, 2009
- Macdonald,A: A Survey of Geometric Algebra and Geometric Calculus, http://faculty.luther.edu/~macdonal, 2009
- Advances in Applied Clifford Algebras - Birkhauser-Springer journal [http://www.clifford-algebras.org]
- Digital papers repository of WSCG 1992 available from http://www.wscg.eu

SIGGRAPH Asia 2012

Questions

???

Contact
Vaclav Skala
c/o University of West Bohemia
Faculty of Applied Sciences
Dept. of Computer Science and Engineering
CZ 306 14 Plzen, Czech Republic
http://www.VaclavSkala.eu skala@kiv.zcu.cz subj. SIGGRAPH Asia

Supported by the Ministry of Education of the Czech Republic, projects ME10060, LA10035, LH12181