Robust Computation in Engineering, Geometry and Duality

ICONS 2012

Reunion Island

Courtesy of http://www.google.cz

Václav Skala

University of West Bohemia, Plzen, Czech Republic
VSB-Technical University, Ostrava, Czech Republic

http://www.VaclavSkala.eu

Plzen (Pilsen) City

Plzen is an old city [first records of Plzen castle 976] city of culture, industry, and brewery.

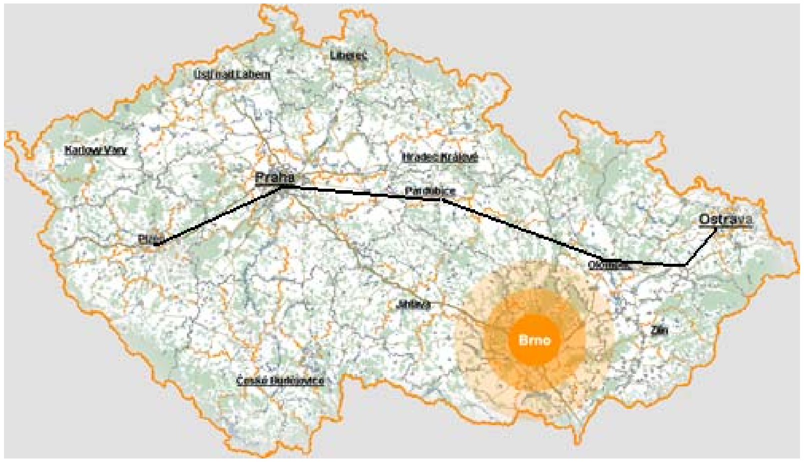
City, where today's beer fermentation process was invented that is why today's beers are called Pilsner - world wide

Ostrava City

Ostrava is

- an industrial city of coal mining & iron melting
- 3rd largest city

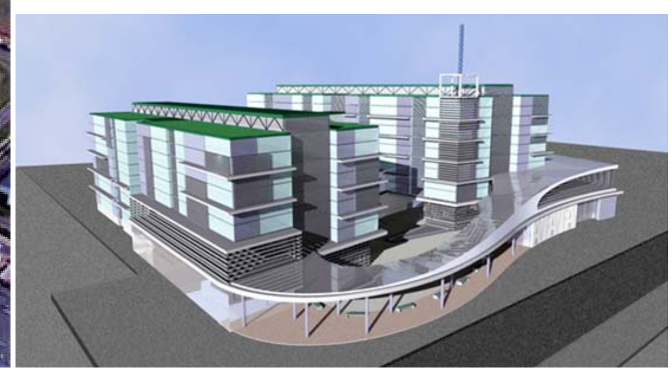


University of West Bohemia 17530 students + 987 PhD students

Computer Science and Engineering Mathematics (including Geomatics)

Physics Cybernetics Mechanics (Computational)

- Over **50%** of income from research and application projects
- NTIS project (investment of 27 mil. EUR)
- 2^{nd} in the ranking of Czech technical / informatics faculties 2009, 2012



“Real science” in the XXI century

Courtesy of Czech Film, Barrandov

History of Mathematics

- Euclid - synthetic geometry 300 BC
- Descartes - analytic geometry 1637
- Gauss – complex algebra 1798
- Hamilton – quaternions 1843
- Grassmann – Grasmann Algebra 1844
- Cayley – Matrix Algebra 1854
- Clifford – Clifford algebra 1878
- Gibbs – vector calculus 1881
- Sylvester – determinants 1878
- Ricci – tensor calculus 1890
- Cartan – differential forms 1908
- Dirac, Pauli – spin algebra 1928

Main line

Hestenes – Space-time algebra 1966 ➔ Geometry Algebra 1984

Robust Computation in Engineering, Geometry and Duality

- Typical engineering and geometrical problems
- Euclidean and projective spaces
- Duality, property of dual transformation
- Algorithm complexity and dual problems
- Robustness & influence to algorithm design
- Typical examples of duality applications

Numerical systems

- Binary system is used nearly exclusively
- Octal & hexadecimal representation is used
- If we would be direct descendants of tetrapods – we would have a great advantage – “simple counting”

Courtesy Clive "Max" Maxfield and Alvin Brown

The first tetrapods had eight fingers on each hand

Figure 1

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

IEEE 758-2008 standard

Mathematically perfect algorithms fail due to instability

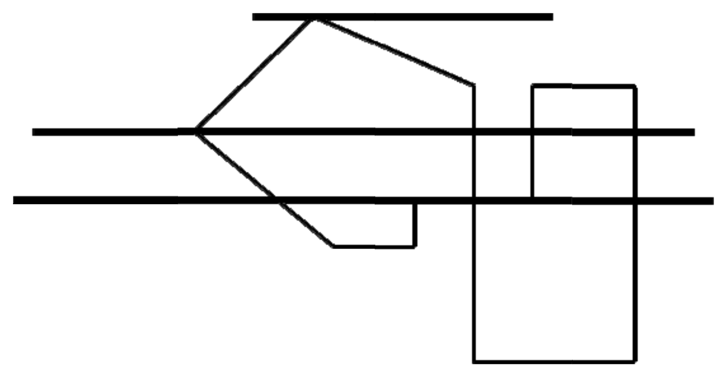
- Main issues
  - stability, robustness of algorithms
  - acceptable speed
  - linear speedup – results depends on HW, CPU …. parameters !
- Numerical stability
  - limited precision of **float** / **double**
  - tests A ? B with **float**s
    - **if** A = B **then** ….. **else** ….. ; **if** A = 0 **then** ….. **else** ….
      should be forbidden in programming languages
  - division operation should be removed or postponed to the last moment if possible - “blue screens”, system resets

Typical examples of instability

- intersection of 2 lines in E3,
- point lies on a line in E2 or a plane in E3

$$Ax + By + C = \mathbf{0} \quad \text{or} \quad Ax + By + Cz + D = \mathbf{0}$$

- detection if a line intersects a polygon, touches a vertex or passes through

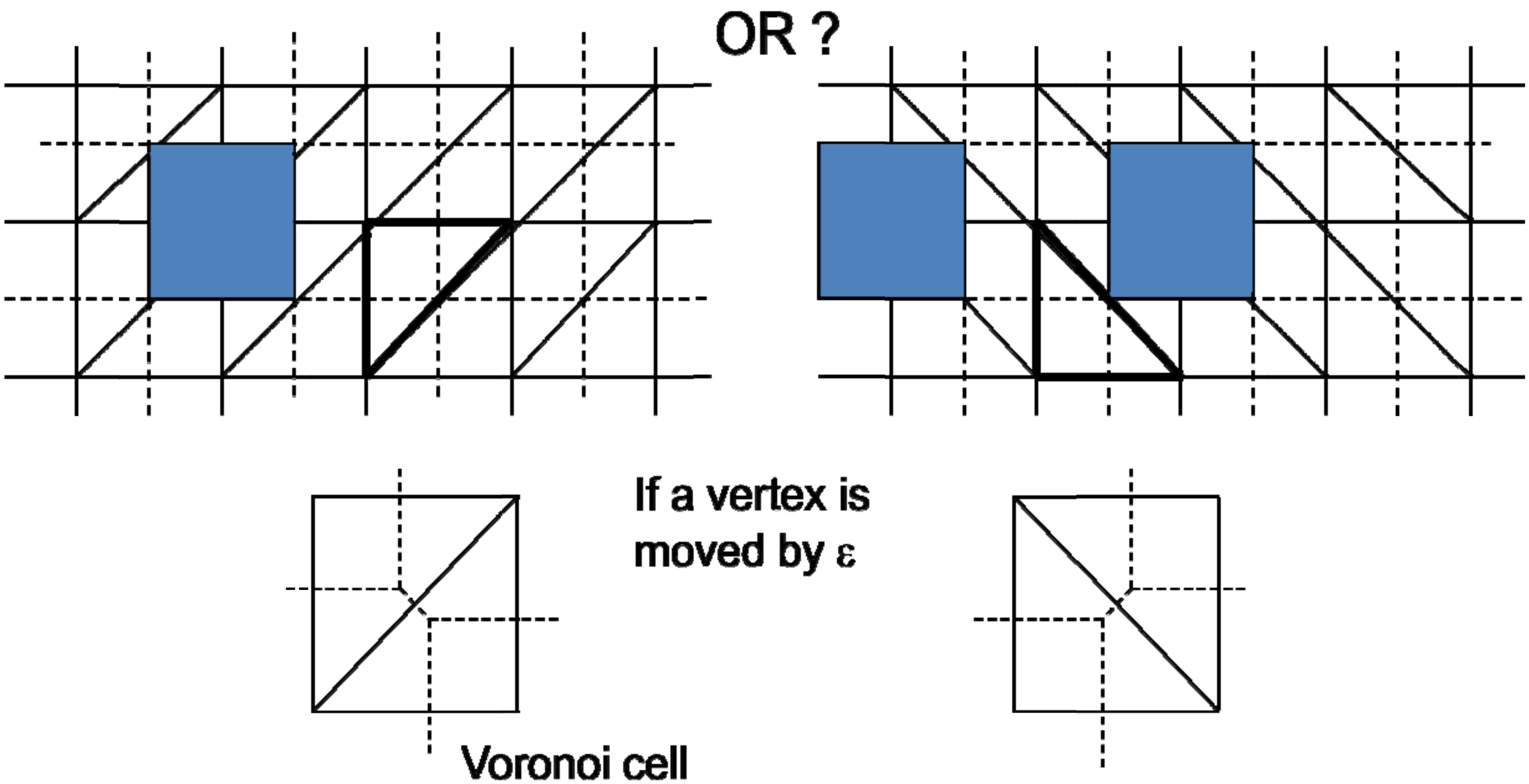


Typical problem

```
double x = -1; double p =  ....;
while ( x < +1)
{     if (x == p) Console.Out.WriteLine(" *** ")
      x += p;
}
##       if p = 0.1 then no output,   if p = 0.25 then expected output
```

Delaunay triangulation & Voronoi diagram

Point inside of a circle given by three points — problems with meshing points in regular rectangular grid.



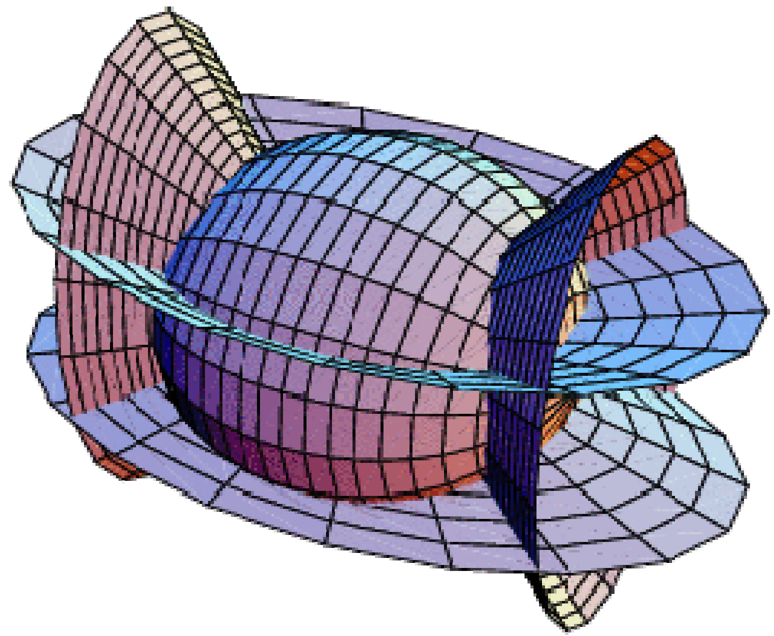
It can be seen that the DT & VD is **very sensitive** to a point position change

?? **ROBUSTNESS** ??

Vectors and Points in Geometry

Vectors – movable, no fixed position

Points – no size, position fixed in the GIVEN coordinate system



Coordinate systems:

- Cartesian – left / right handed
  right handed system is used
- Polar
- Spherical
- and many others, e.g. Confocal Ellipsoidal Coordinates (http://mathworld.wolfram.com/ ConfocalEllipsoidalCoordinates.html)

Vectors and Points in Geometry

Vector representation $\boldsymbol{v} = (v_x , v_y , v_z : 0)$

Point representation $\boldsymbol{P} = (P_x , P_y , P_z : 1)$, resp. $(P_x , P_y , P_z : P_w)$,

Many libraries do not distinguish between points and vectors and treat them in the same manner

!! BE CAREFUL !!

$\boldsymbol{v} = \boldsymbol{P}_1 - \boldsymbol{P}_0 = (P_{x1} , P_{y1} , P_{z1} : 1) - (P_{x0} , P_{y0} , P_{z0} : 1) =$

$(P_{x1} - P_{x0} , P_{y1} - P_{y0} , P_{z1} - P_{z0} : 0) = (v_x , v_y , v_z : 0)$

!!! Do not make it on CPU/GPU – result $(v_x , v_y , v_z : \varepsilon)$

It is a point $(v_x / \varepsilon, v_y /\varepsilon, v_z / \varepsilon)$ in E^3

Floating point

- Not all numbers are represented correctly
- Logarithmic arithmetic
- Continuous fractions
- Interval arithmetic

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

$$\pi = [3; 7,15,1,292,1,1,1,2,1,3,1 \ldots]$$

- NOT valid identities

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

$$\mathrm{x}^2 - \mathrm{y}^2 = (x - y)(x + y)$$

x + y = [a + c, b + d]   x = [a , b]

x - y = [a - d, b - c]   y = [c , d]

x × y = [min(ac, ad, bc, bd), max(ac, ad, bc, bd)]

x / y = [min(a/c, a/d, b/c, b/d),

max(a/c, a/d, b/c, b/d)] if y ≠ 0

Statements like

if <float> = <float> then .... or **if <float> ≠ <float> then** ....

should not be

Quadratic equation

$$at^2 + bt + c = 0$$

If $b^2 \gg 4ac$ then

$$q = -(b + sign(b)\sqrt{b^2 - 4ac})/2$$

$$t_1 = {}^{q}/_{a} \qquad t_2 = {}^{c}/_{a}$$

to get more reliable results.

Function value computation at $x = 77617,\ y = 33096$

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

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

$f = 1{,}1726039400532$ double precision

$f = 1{,}172603940053178631858834904520183$ extended precision

The correct result is "somewhere" in the interval of

$$[-0{,}827396059946821368141165095479816292005,$$
$$-0{,}827396059946821368141165095479816291986]$$

Exact solution

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

Summation is one of often used computations.

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

or

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

The result should be one. The correctness in summation is very important in power series computations. !!!!ORDER of summation

$$\sum_{n=1}^{10^6} \frac{1}{n} = 14.357357 \qquad \sum_{n=10^6}^{1} \frac{1}{n} = 14.392651$$

Recursion

- Towers o Hanoi

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

- Ackermann function

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

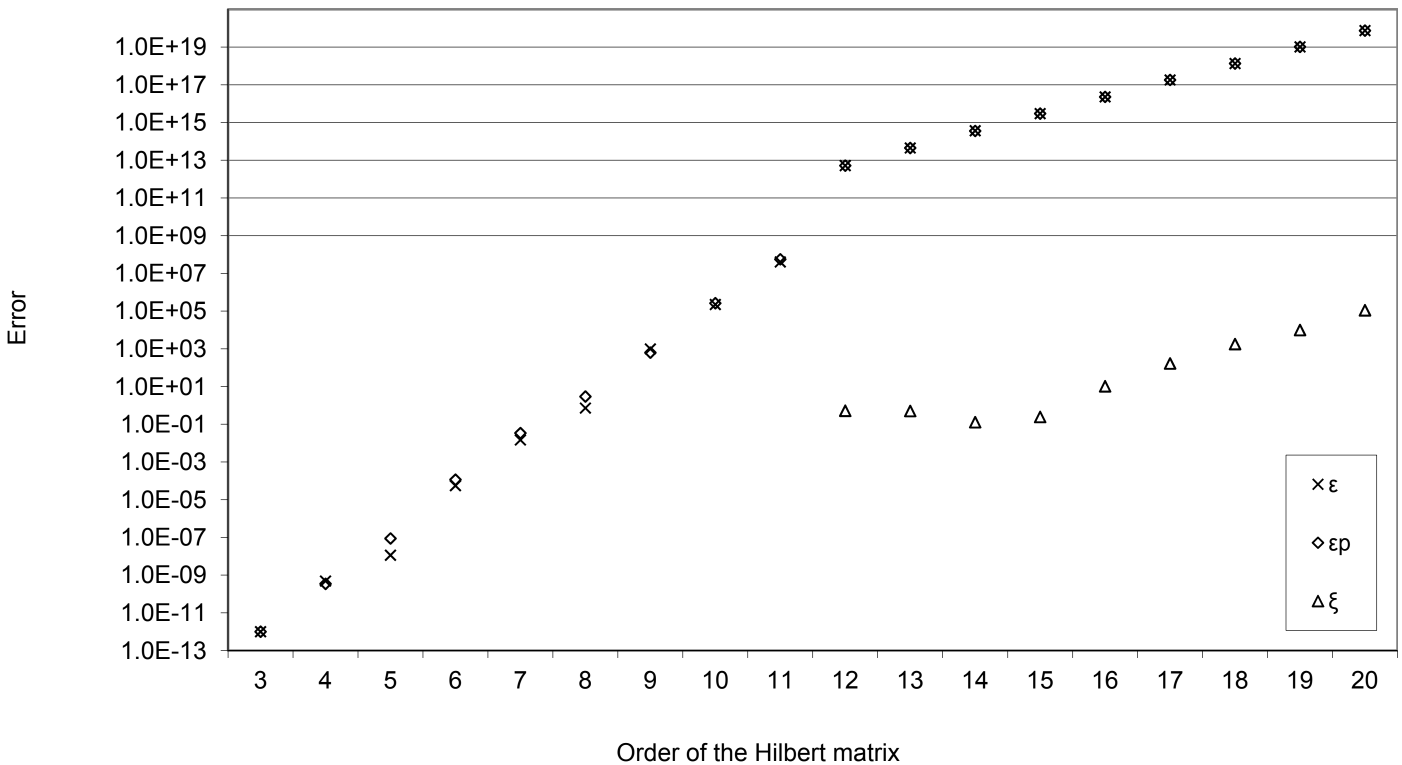
The value of the function grows very fast as

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

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

Hilbert's Matrix $\boldsymbol{H}_{ij} = \frac{1}{i+j-1}$

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



Mathematical "forms" There are several "forms":

- **Implicit**

  $F(x, y, z) = 0$ $F(\boldsymbol{x}) = 0$ $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$

  there is no orientation, e.g.
  - if $F(\boldsymbol{x}) = 0$ is a iso-curve there is no hint how to find another point of this curve, resp. a line segment approximating the curve => tracing algorithms
  - if $F(\boldsymbol{x}) = 0$ is a iso-surface there is no hint how to find another point of this surface => iso-surface extraction algorithms

- **Parametrical** – $\boldsymbol{x} = \boldsymbol{x}(u)$ $\boldsymbol{x} = \boldsymbol{x}(u, v)$

  points of a curve are "ORDERED" according to a parameter a

- **Explicit**

  $z = f(x)$ $z = f(x, y)$

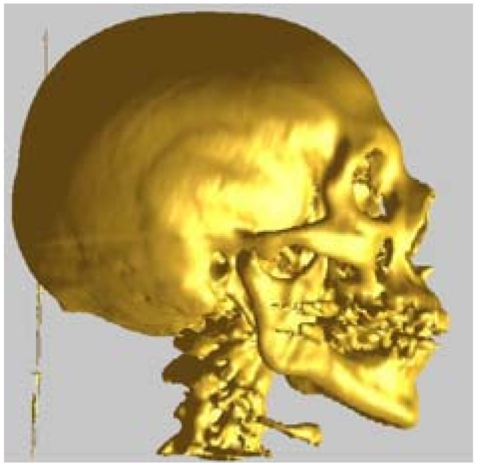
  for the given values x, resp. x, y we get function value z

Implicit form

- Is used for separation – plane or for detection if a point is inside or outside, e.g. a circle etc.
- There is always a question how to compute **complexity** of computations × **precision** of computation
- Compiler optimization is **DANGEROUS** in general

$$\begin{vmatrix} A_x & A_y, & A_x^2 + A_y^2, & 1 \\ B_x & B_y, & B_x^2 + B_y^2, & 1 \\ C_x & C_y, & C_x^2 + C_y^2, & 1 \\ D_x & D_y, & D_x^2 + D_y^2, & 1 \end{vmatrix} = \begin{vmatrix} A_x - D_x, & A_y - D_y, & (A_x^2 - D_x^2) + (A_y^2 - D_y^2) \\ B_x - D_x, & B_y - D_y, & (B_x^2 - D_x^2) + (B_y^2 - D_y^2) \\ C_x - D_x, & C_y - D_y, & (C_x^2 - D_x^2) + (C_y^2 - D_y^2) \end{vmatrix} > 0$$

Data processing - main field in computer science



Data processing itself can be split to two main areas:

- **processing of textual data**

  limited interval of values, unlimited dimensionality

  [char as one dimension -

  Methionylthreonylthreonylglutaminylarginyl...isoleucine 189,819 chars]

  - no interpolation is defined

- **processing of numerical data**

  unlimited interval of values,

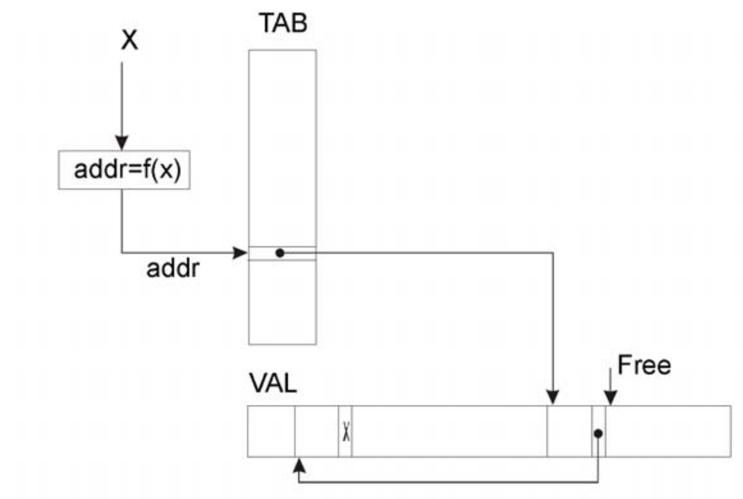
  limited dimensionality – usually 2 or 3

  - interpolation **can be used**

	Textual	Graphical
Dim	∞	2, 3
Interval	0-255 (ASCII)	$(-\infty, \infty)$

Hash functions

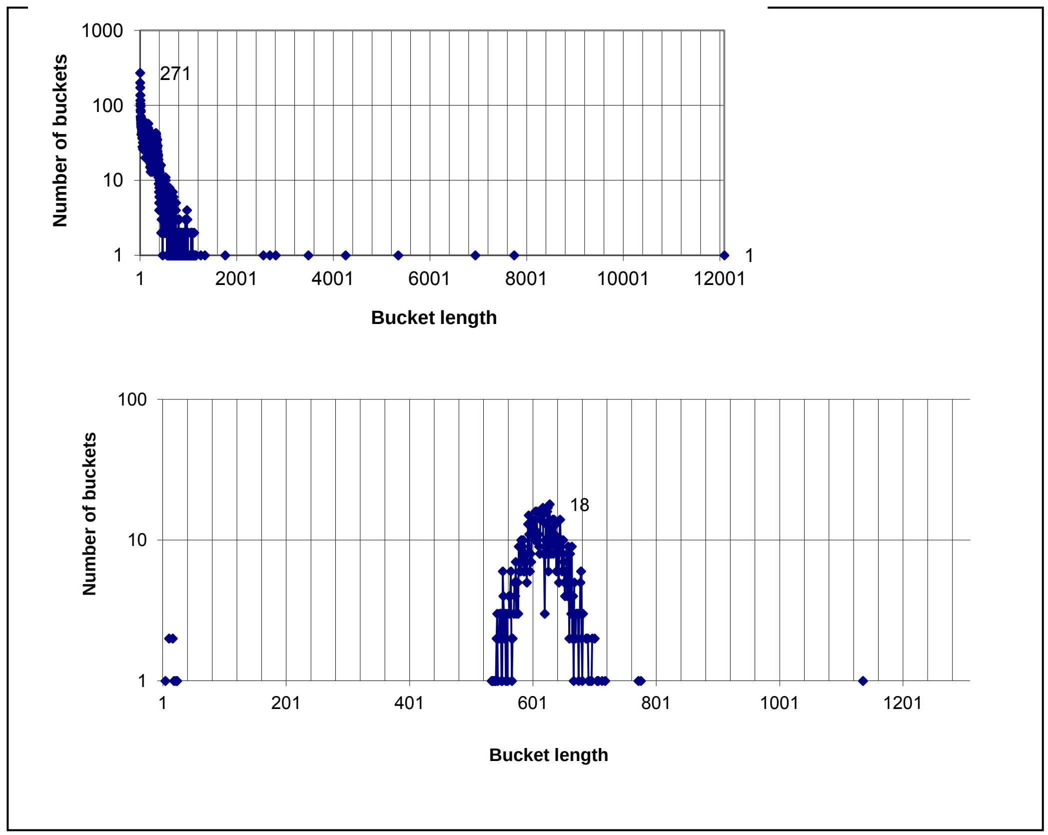
- usually used for textual data processing
- prime numbers and modulo operations are used for the hash function

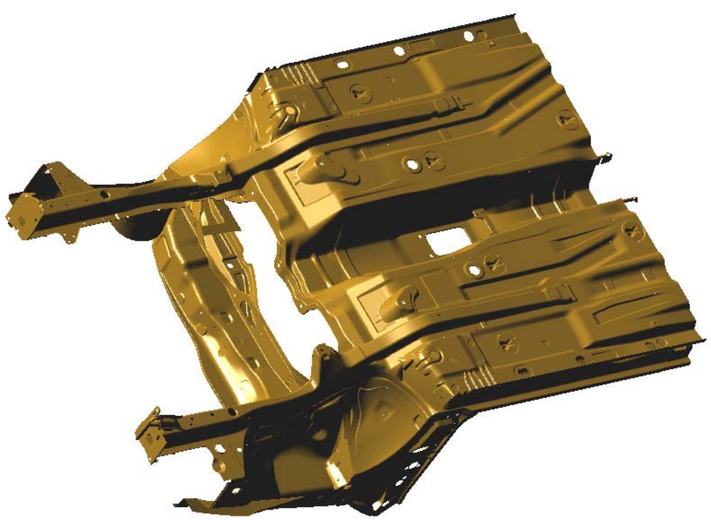


Usual form

$$Addr = \lfloor 3x + 5y + 7z \rfloor \, \mathbf{mod} \; size$$

multiplication int * float is needed

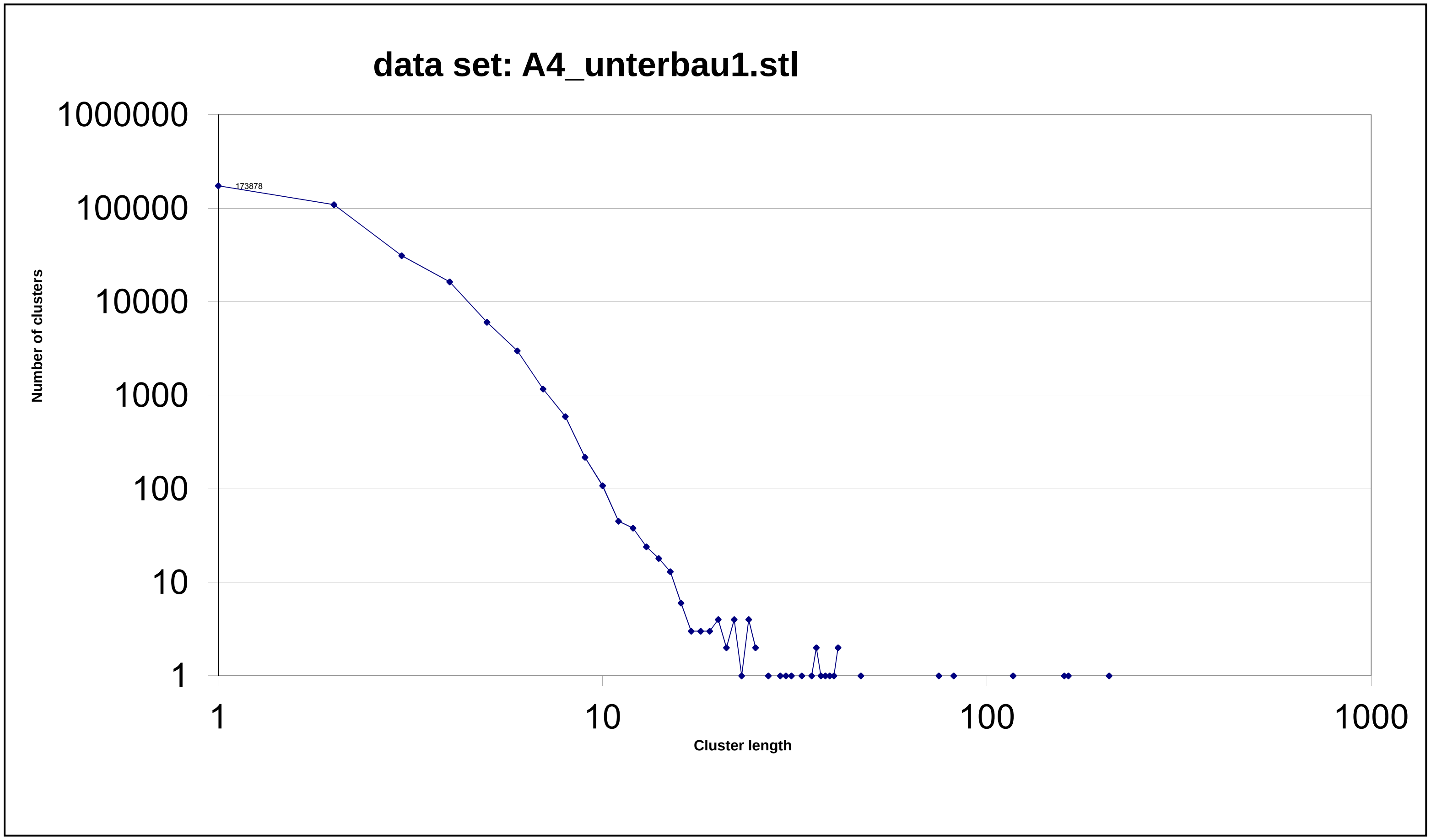




If the hash function is constructed as

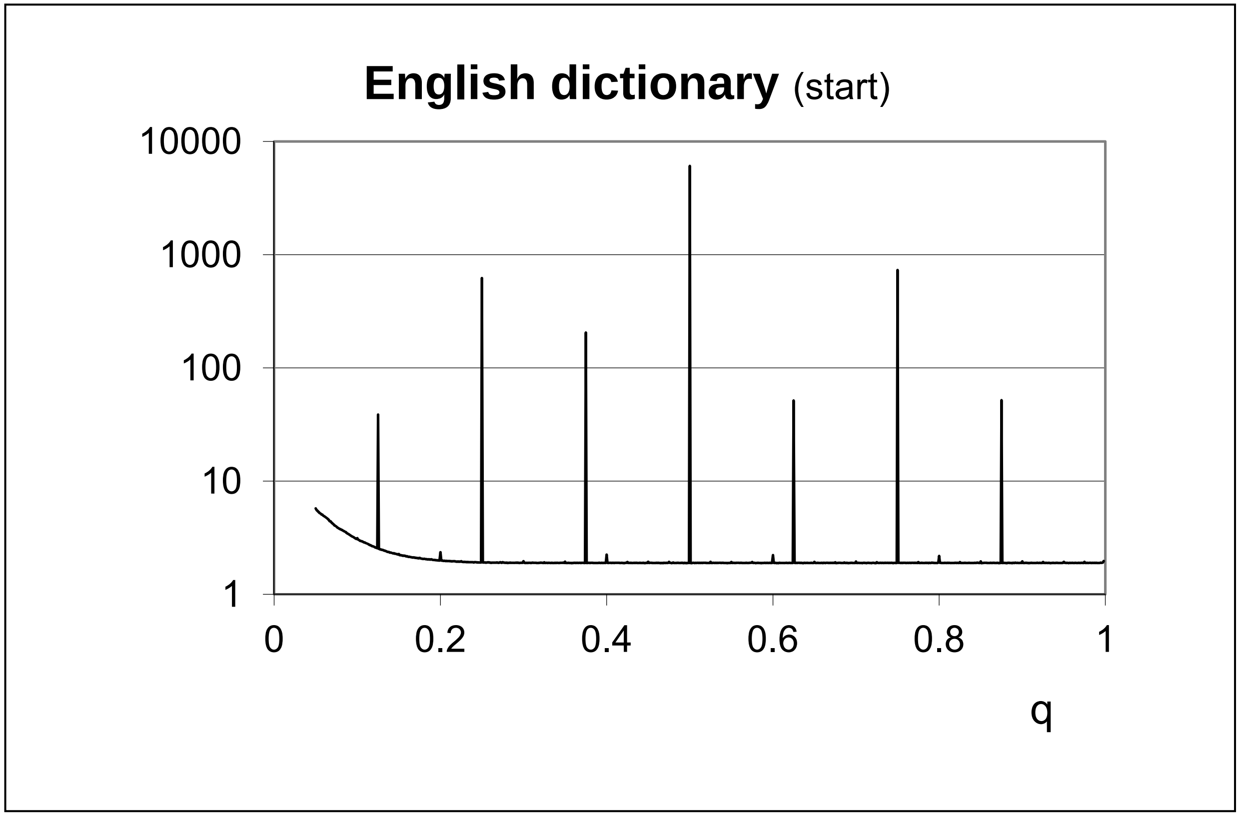
$$Addr = \lfloor \alpha x + \beta y + \gamma z \rfloor \mathbf{mod}\, m$$

where α, β, γ are "irrational" numbers and $m = 2^k - 1$ better distribution is



obtained => much faster processing.

Textual processing



The has function is constructed as

$$h(\boldsymbol{x}) = \left(C * \sum_{i=1}^{L} q^i x_i \right) \boldsymbol{mod}\; m$$

q „irrational“ $0 < q < 1$

$m = 2^k - 1$

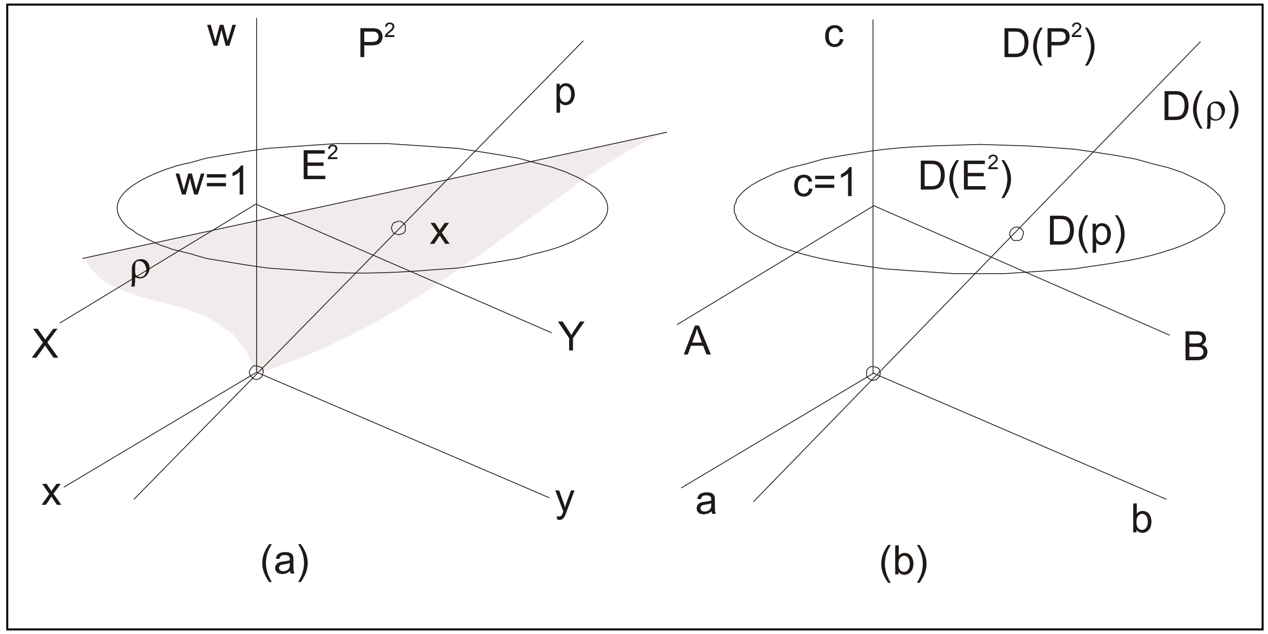
Both geometrical and textual hash function design have the same approach coefficients are “irrational” and no division operation is needed.

Some differences for Czech, Hebrew, English, German, Arabic, … languages and “chemical” words.

Projective Space

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

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



Conversion:

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

$\& \; w \neq 0$

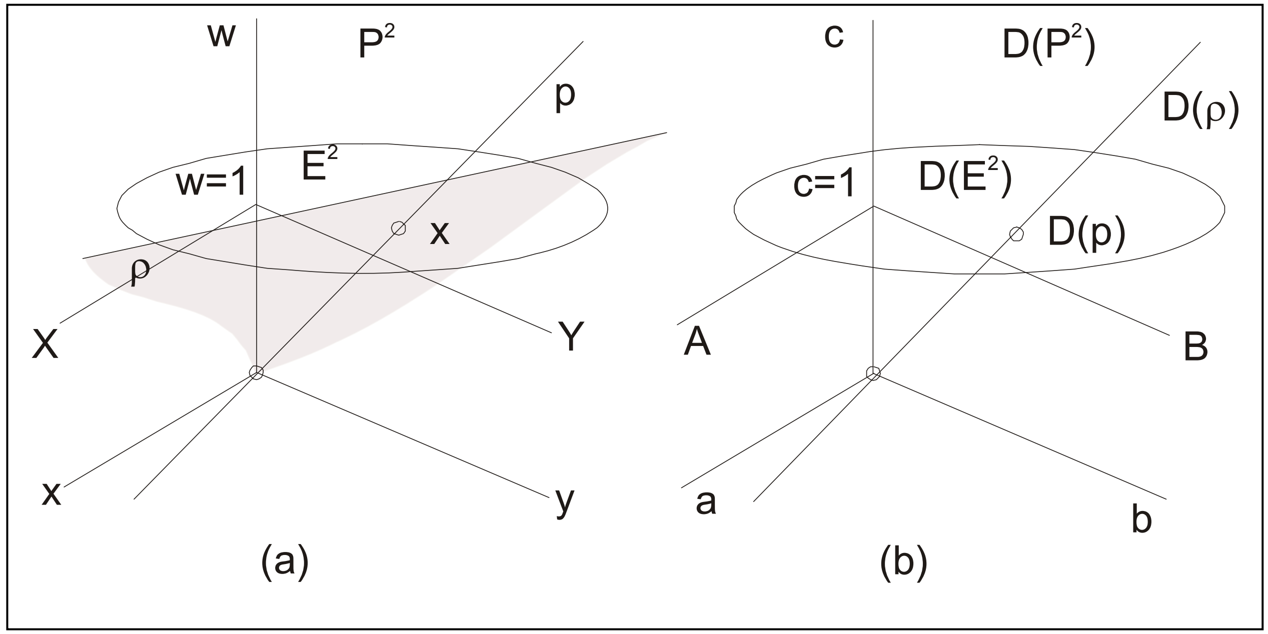
If $w = 0$ then $\boldsymbol{x}$ represents "an ideal point" - a point in infinity, i.e. it is a directional vector.

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

Duality

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

$$aX + bY + c = 0$$



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

$$ax + by + cw = 0 \quad \text{i.e.}$$

$$\boldsymbol{p}^T\boldsymbol{x} = 0 \qquad \boldsymbol{p} = [\,a, b: c]^T \qquad \boldsymbol{x} = [\,x, y: w]^T$$

It is actually a plane in the projective space P^2 (point $[0, 0: 0]^T$ excluded), i.e. line $p \in E^2$: $\boldsymbol{p} = [a, b: c]^T$

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.

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:

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

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

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

Proof 1

Let the line $\boldsymbol{p} \in E^2$ be defined in homogeneous coordinates as

$$ax + by + cw = 0$$

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

$$\mathbf{p}^T \mathbf{x}_1 = 0 \quad \text{and} \quad \mathbf{p}^T \mathbf{x}_2 = 0 \qquad \text{where: } \mathbf{p} = [a,b,c]^T$$

It means that any point $\mathbf{x}$ 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 ***p*** vector is defined as

$$\mathbf{p} = \det\begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}$$

We can write $(\mathbf{x}_1 \times \mathbf{x}_2)^T \mathbf{x} = 0$ i.e. $\det\begin{bmatrix} x & y & w \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix} = 0$

Evaluating the determinant

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

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

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

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

No division operation !

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!

Computation in Projective Space

	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 $\boldsymbol{p}' = \frac{1}{det(\boldsymbol{T})}(\boldsymbol{T}^{-1})^T\boldsymbol{p}$, resp. $\boldsymbol{\rho}' = \frac{1}{det(\boldsymbol{T})}(\boldsymbol{T}^{-1})^T\boldsymbol{\rho}$

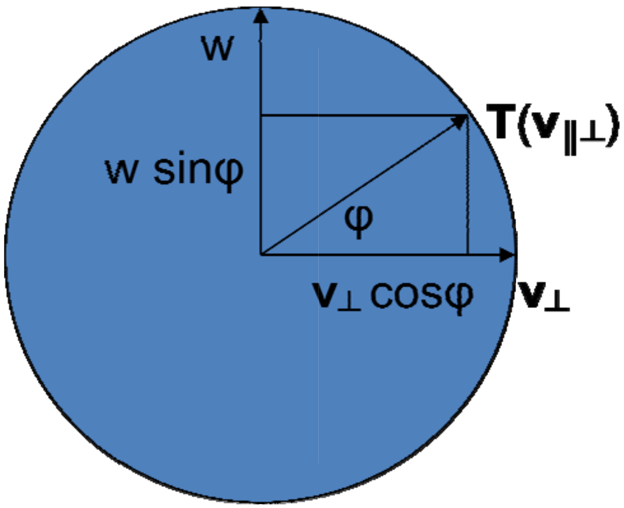
As the computation is made **in the projective space** we can write

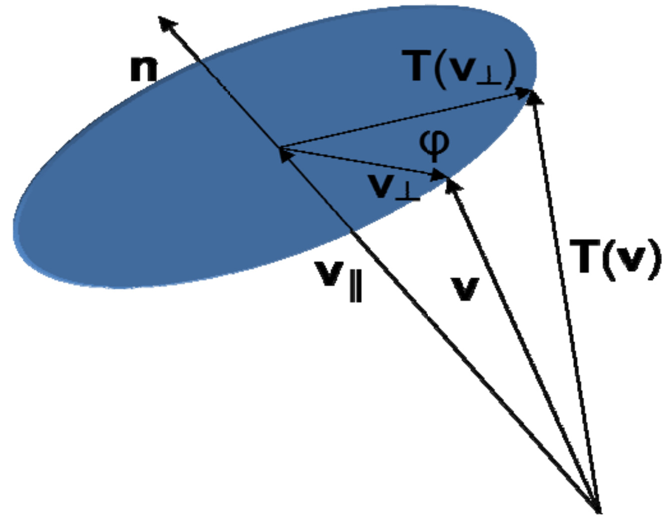
$$\boldsymbol{p}' = (\boldsymbol{T}^{-1})^T\boldsymbol{p} \text{ , resp. } \boldsymbol{\rho}' = (\boldsymbol{T}^{-1})^T\boldsymbol{\rho}$$

THIS IS SIGNIFICANT SIMPLIFICATION OF COMPUTATIONS

Computation in Projective Space

Transformation about a general axis





Computation in Projective Space

Transformation about a general axis

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

$$\|\boldsymbol{n}\| = 1 \qquad \boldsymbol{n} \otimes \boldsymbol{n} = \boldsymbol{n}^T \boldsymbol{n} \qquad \text{where: } \boldsymbol{W}\boldsymbol{v} = \boldsymbol{w} \times \boldsymbol{v}$$

Instead of usually used transformations:

$$\boldsymbol{Q} = \boldsymbol{T}^{-\mathbf{1}} \boldsymbol{R}_{xy}^{-1} \boldsymbol{R}_{yz}^{-1} \boldsymbol{R}(\vartheta) \boldsymbol{R}_{zy} \boldsymbol{R}_{xy} \boldsymbol{T}$$

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

Computation in Projective Space

Interpolation

Linear parametrization

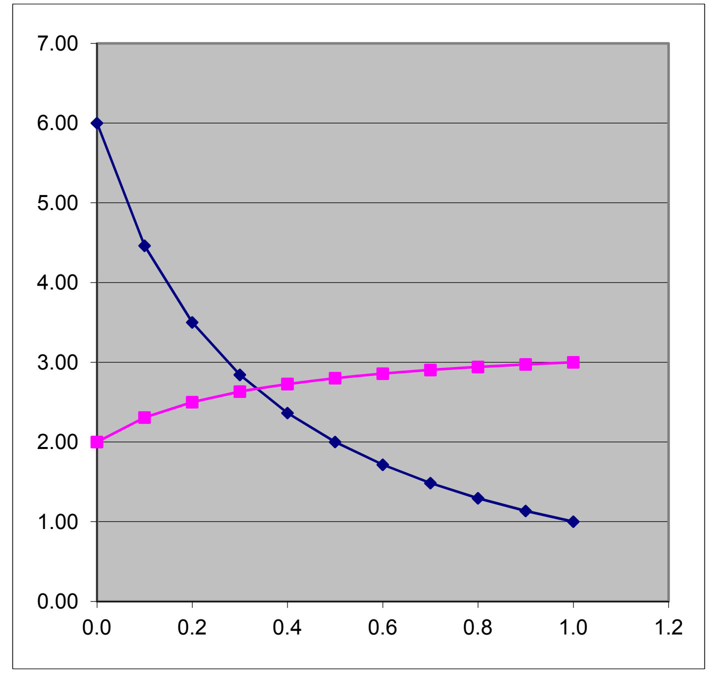
$$\mathbf{X}(t) = \mathbf{X}_0 + (\mathbf{X}_1 - \mathbf{X}_0)t \quad t \in (-\infty, \infty)$$

Non-linear (monotonous) parametrization

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

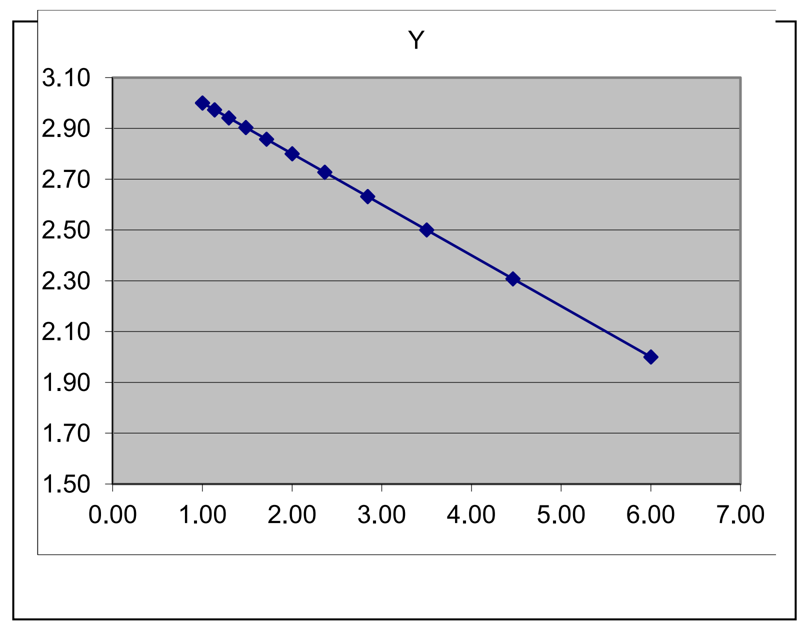
$$x(t) = x_0 + (x_1 - x_0)t \qquad y(t) = y_0 + (y_1 - y_0)t$$

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

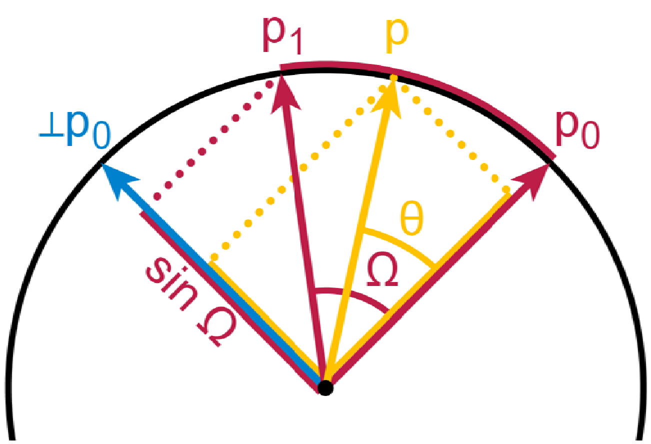


- It means that we can interpolate using homogeneous coordinates without a need of "normalization" to E^k !!
- Homogeneous coordinate w ≥ 0

In many algorithms, we need "monotonous" parameterization, only



Computation in Projective Space



/ikipedia

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 generally incorrect**! $\left[\frac{0}{0}\right]$

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

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

Homogeneous coordinate

Computation in Projective Space

Intersection line – plane

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

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

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

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

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

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

TEST

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

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

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

Line Clipping

procedure CLIP_L; {Skala – Vol.21, No.11, pp.905-914}

{ $\boldsymbol{x}_A$, $\boldsymbol{x}_B$ – 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**];

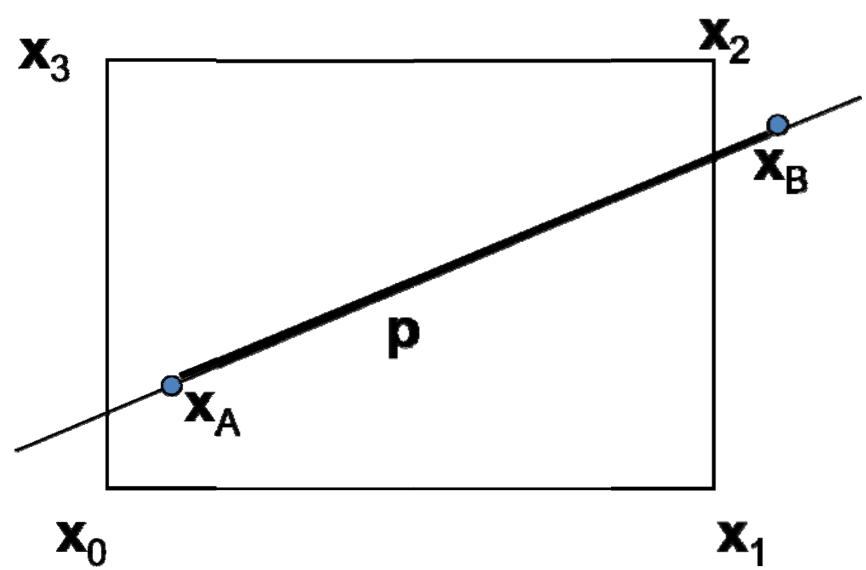
{6} $\boldsymbol{x}_A := \boldsymbol{p} \times \boldsymbol{e}_i$;$\boldsymbol{x}_B := \boldsymbol{p} \times \boldsymbol{e}_j$;

{7} **DRAW** ($\boldsymbol{x}_A$; $\boldsymbol{x}_B$) {$\boldsymbol{e}_i$ – *i-th edge* }

end {CLIP_L};

Line clipping in E^2 algorithms

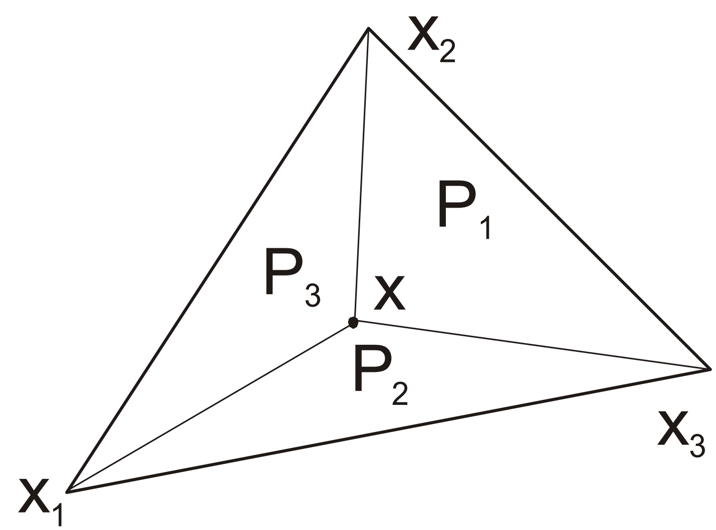
- Cohen-Sutherland
- Liang-Barsky
- Hodgman
- Skala – modification of Clip_L for line segments



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

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

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

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”

Computation in Projective Space

$$b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X = 0$$
$$b_1 Y_1 + b_2 Y_2 + b_3 Y_3 + b_4 Y = 0$$
$$b_1 + b_2 + b_3 + b_4 = 0$$
$$b_i = -a_i b_4 \quad i = 1, \ldots, 3 \qquad b_4 \neq 0$$

Rewriting

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

$$\mathbf{b} = \boldsymbol{\xi} \times \boldsymbol{\eta} \times \mathbf{w}$$
$$\mathbf{b} = [b_1, b_2, b_3, b_4]^T$$
$$\boldsymbol{\xi} = [X_1, X_2, X_3, X]^T$$
$$\boldsymbol{\eta} = [Y_1, Y_2, Y_3, Y]^T$$
$$\mathbf{w} = [1, 1, 1, 1]^T$$

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 $w_i \neq 1$

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

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

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

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

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

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

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

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

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

=> **entities**:

projective scalar, projective vector

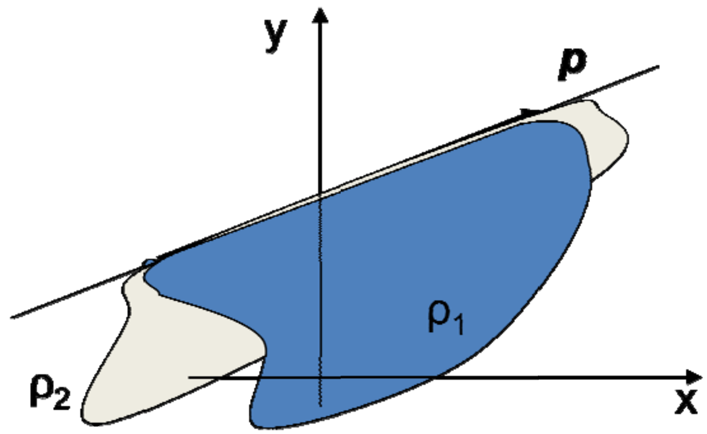
(Skala,V.: Barycentric coordinates computation in homogeneous coordinates, Computers&Graphics, 2008)

Computation in Projective Space

Line in E3 as Two Plane Intersection

- Plücker's coordinates – **complicated** computation

- Projective computation



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

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

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

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

Computation in Projective Space

Standard formula

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

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

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

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

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

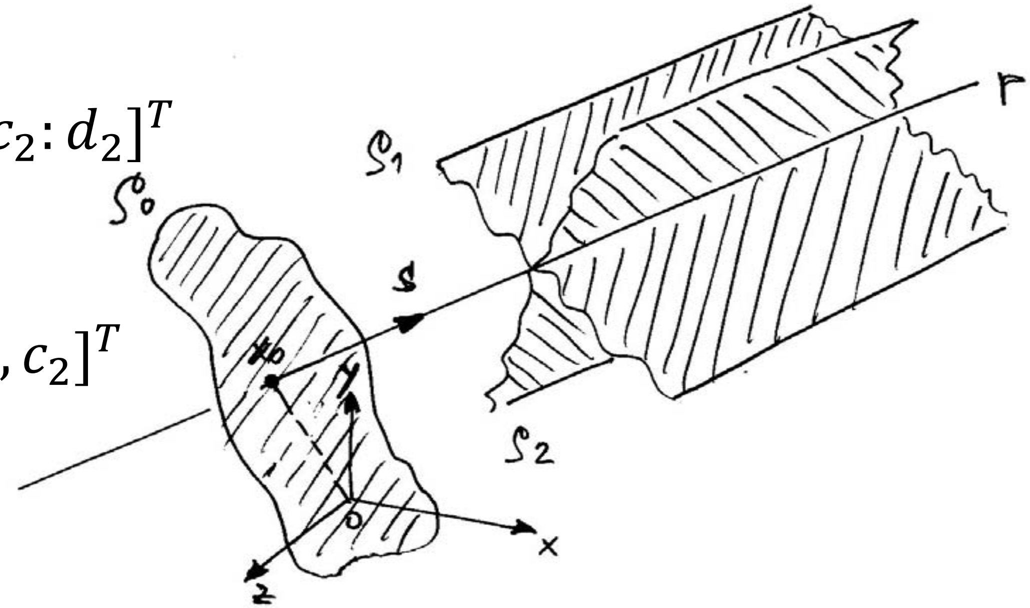
normal vectors are

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

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

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

Computation in Projective Space

Advantages

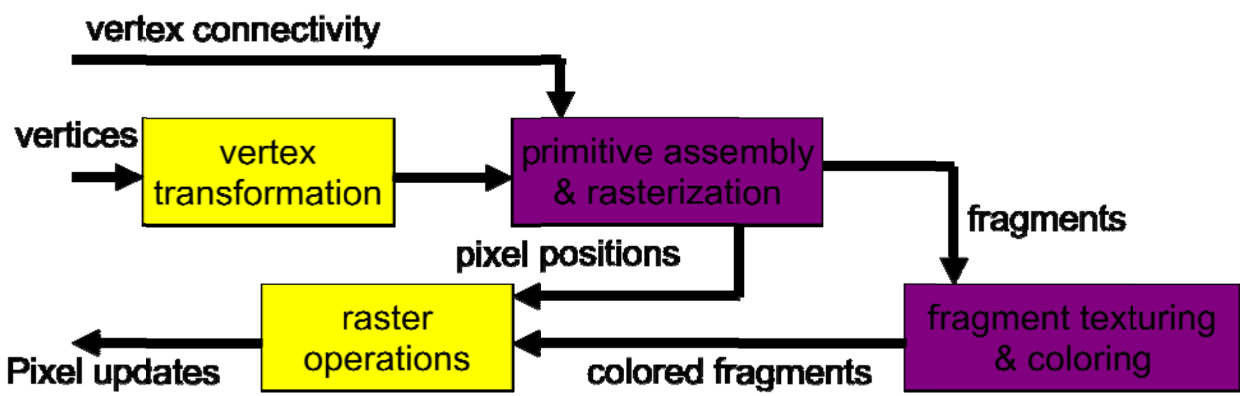
- "infinity" is well represented
- No division operation is needed
- Many mathematical formula are simpler
- One code sequence solve primary and dual problems
- Supports matrix – vector operations in hardware – like GPU etc.

Disadvantages

- Careful handling with formula as the projective space
- Exponents of the homogeneous vector can overflow – it should be normalized not supported by the current hardware – P_Lib use on GPU (C# and C++)

Computation in Projective Space

- 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



Computation in Projective Space

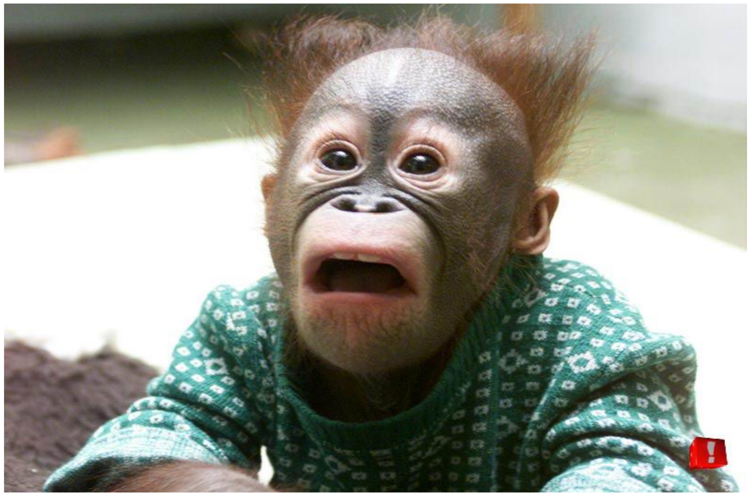
4D cross product can be implemented in Cg/HLSL on GPU (not optimal implementation) as:

```
float4 cross_4D(float4 x1, float4 x2, float4 x3)
  { float4 a;
    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;
  }
```

Acknowledgment

Thanks belong to colleagues at UWB & VSB universities for discussions and help, to many authors of published related papers, I haven't refer properly (a list would be too long*).
Research supported by the MSMT CR,
projects No. LA10035, ME10060

Questions



Contact: Vaclav Skala http://www.VaclavSkala.eu
University of West Bohemia, Plzen & VSB-Technical University, Ostrava
Czech Republic

*) Please, find related reference in:

- Skala,V.: Geometry, Duality and Robust Computation in Engineering, submitted for publication, 2012
- Skala,V.: Interpolation and Intersection Algorithms and GPU, ICONS2012, VisGra 2012 workshop, accepted for publications, 2012

Main References

[1] van Verth,J.M., Bishop,L.M.: Essential Mathematics for Games and Interactive Applications, Morgan Kaufmann 2005

[2] Skala,V.: GPU Computation in Projective Space Using Homogeneous Coordinates , Game Programming GEMS 6 (Ed.Dickheiser,M.), pp.137-147, Charles River Media, 2006

[3] 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

[4] Skala,V.: Barycentric coordinates computation in homogeneous coordinates, submitted for publication, 2007

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

[6] Yamaguchi,F.: Computer Aided Design – A totally Four-Dimensional Approach, Springer Verlag 2002

[7] Fernando,R., Kilgard,M.J.: The Cg Tutorial, Addison Wesley, 2003

[8] Skala,V., Kaiser,J., Ondracka,V.: Library for Computation in the Projective Space, Aplimat 2007 conf., 2007

[9] Uhlir,K., Skala,V.: Radial Basis Functions, EUSIPCO 2005

RESOURCES FOR ROBUST COMPUTATION

Numerical nonrobustness causes all kinds of failures. But can you produce an example with an infinite loop? This and other forms of manifestation are discussed in Anatomy of Algorithmic Failures. It is intended to provide classroom examples. Source code available.

Robust Geometric Algorithms and their Implementation, Guest Editorial Forward from a Special Issue of Computational Geometry: Theory and Applications (CGTA 33:1, 2006).

Resource Page for ``Survey/Tutorial on Exact Geometric Computing'' Lectures at Workshop on Geometric Computing, University of Hong Kong, June 27--29, 2001.

General Forums on Nonrobustness Issues

-- DIMACS Workshop on Implementation of Geometric Algorithms Dec 4-6, 2002
-- SIAM Minisymposium: Robust Geometric Computation (2001)
-- MSRI Workshop on Foundations of CAD (1999)
-- SIAM Workshop on Integration of CAD and CFD (1999)
-- Emerging Challenges in Computational Topology Report (1999)
--Computational Geometry Task Force Report (1996)
-- SIGGRAPH'98 Panel on Robust Geometry. Position Statements from panelists.
--ACM Strategic Directions Report (1996)
--Tom Peters's presentation on Non-robustness Issues in CAGD [here is a local copy] This work is part of the NSF/DARPA CARGO Program (2001-4) in which non-robustness and topological consistency issues are addressed.

Non-robustness in the News

-- Patriot Missile Defense Saga, from US General Accounting Office (local copy)
-- Ariane 5 Saga, from European Space Agency
-- Software Bugs Cost US economy $59.5 billion/year. The report focused on the financial sector, and the automotive and aircraft manufacturing industries. In the latter 2 industries, the cost is estimated at $1.8 billion/year. Report was prepared by Research Triangle Institute for NIST.
-- Disasters attributable to numerical errors, from Doug Arnold (including the North Sea oil rig collapse)

Projects and Groups

--K. Mehlhorn at Max-Planck Institute of Computer Science is involved in various robustness projects (LEDA, CGAL, EXACUS).
--PRISME Project at INRIA, Sophia-Antipolia, directed by J.-D. Boissonnat.
-- Superrobust Computation Project of K. Sugihara at Tokyo University
-- Arenaire Project of J.-M. Muller.
-- iRRAM Project of N. Mueller is a C++ package for error-free real arithmetic based on the concept of a real RAM.
-- Professor Cuyt's Group on Computer Arithmetic and Numerical Techniques.
-- Evaluation of Special Functions from Dan Lozier at NIST: includes a function evaluator service.

RETRIEVED from http://www.cs.nyu.edu/csweb/Research/Groups/ **http://www.cs.nyu.edu/csweb/Research/Groups/**

Software

-- GMP Home. New in GMP 3.1 (FFT Based Multiplication!)

--MPFR homepage The MPFR library is a C library for multiprecision floating-point computations with exact rounding. It is based on the GMP multiprecision library and will replace the MPF class starting with version 3.1 of GMP. Here are some timings .

-- Intel's Open source for Numerics

-- Arithmetic Explorer

Robust Meshing and Triangulation

-- GNU Triangulated Surface (GTS) Library: Open Source free software for 3D surfaces triangular meshes. Version 0.4.0 (Jan 2001).
FEATURES: based on the GTK+ GUI Toolkit, 2D dynamic constrained Delaunay triangulation, robust predicates of Shewchuk, Boolean set operations, multiresolution models, dynamic view-independent LOD, some view-dependent LOD, Kd-trees, collision detection.
[Bibliography | Reference Manual |]

Computing transcendental functions.

-- The Table Maker's Dilemma

-- Elementary Functions, Algorithms and Implementation, book by Jean-Michel Muller from Birkhauser, Boston (1997).

Organizations and Web Resources

-- CCA Net (Computability and Complexity in Analysis)

-- Interval Computation

-- Real Algebraic and Analytic Geometry (RAAG)

-- Foundations of Computational Mathematics

Software Resources

Source Forge: free Open Source developer services (CVS, mailists, forums, site hosting, etc). Thousands of projects in every field.

Conferences

-- IEEE Symposium on Computer Arithmetic. Call for Papers for 15th Symposium on Computer Arithmetic (Nov 1, 2000).

-- 4th Conference on Real Numbers and Computers

-- 4th Workshop on Computability and Complexity in Analysis Sep 17-20, 2000, Swansea, Wales.

Bibliography Collection

-- Constructivity, Computability and Complexity in Analysis, a collection of over 500 entries.

-- Numerical techniques from Rational Approximation Theory and Computer Arithmetic, a collection of over 2500 entries.

RETRIEVED from http://www.cs.nyu.edu/csweb/Research/Groups/ **http://www.cs.nyu.edu/csweb/Research/Groups/**