INF555

Fundamentals of 3D

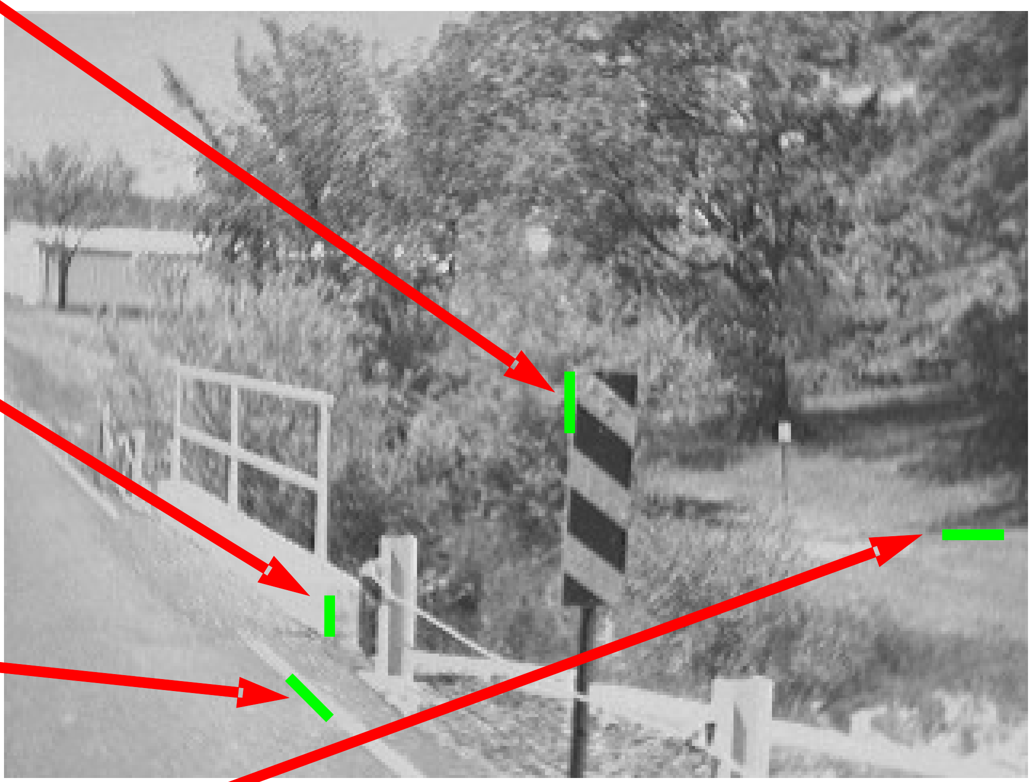
Lecture 3:

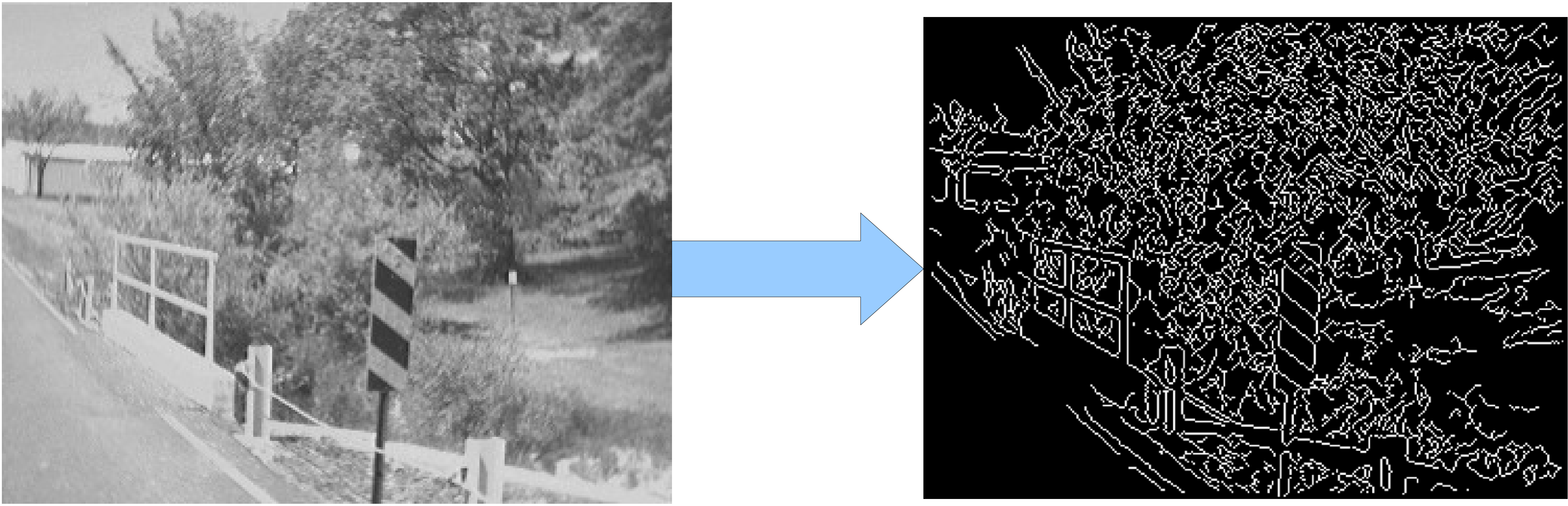
Debriefing: Lecture 2
Rigid transformations
Quaternions
Iterative Closest Point (+Kd-trees)

Frank Nielsen
nielsen@lix.polytechnique.fr

Harris-Stephens' combined corner/edge detector

- Depth discontinuity
- Surface orientation discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)
- Illumination discontinuity (e.g., shadow)





Harris-Stephens edge detector

Aim at finding good feature

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Gradient with respect to x, times gradient with respect to y

Sum over image region – area we are checking for corner

Harris-Stephens edge detector

Measure the corner response as

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

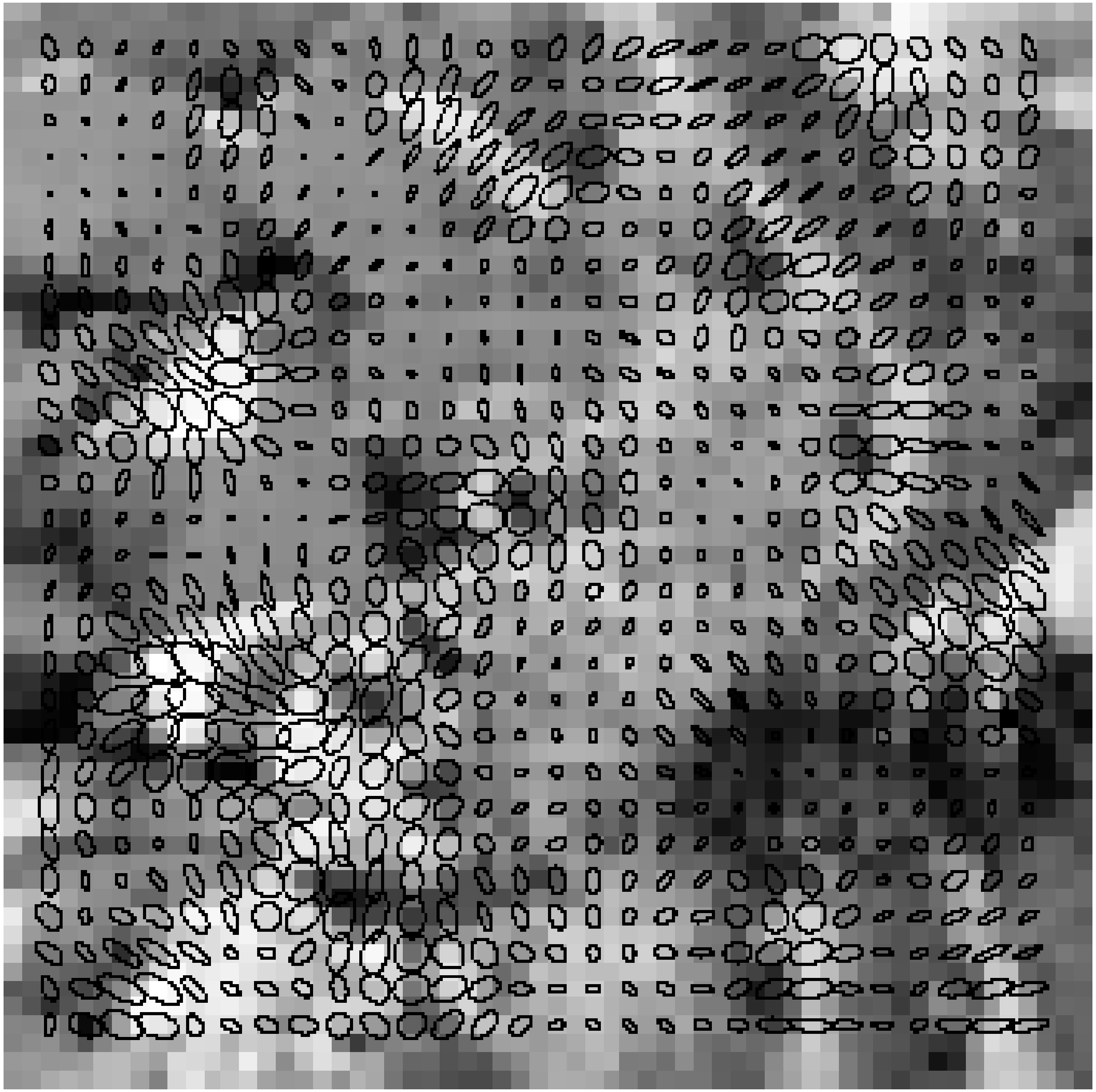
Avoid computing eigenvalues themselves.

(k – empirical constant, k = 0.04-0.06)

Algorithm:

– Find points with large corner response function R (R > threshold)

– Take the points of **local maxima** of R



Edge thresholding hysterisis

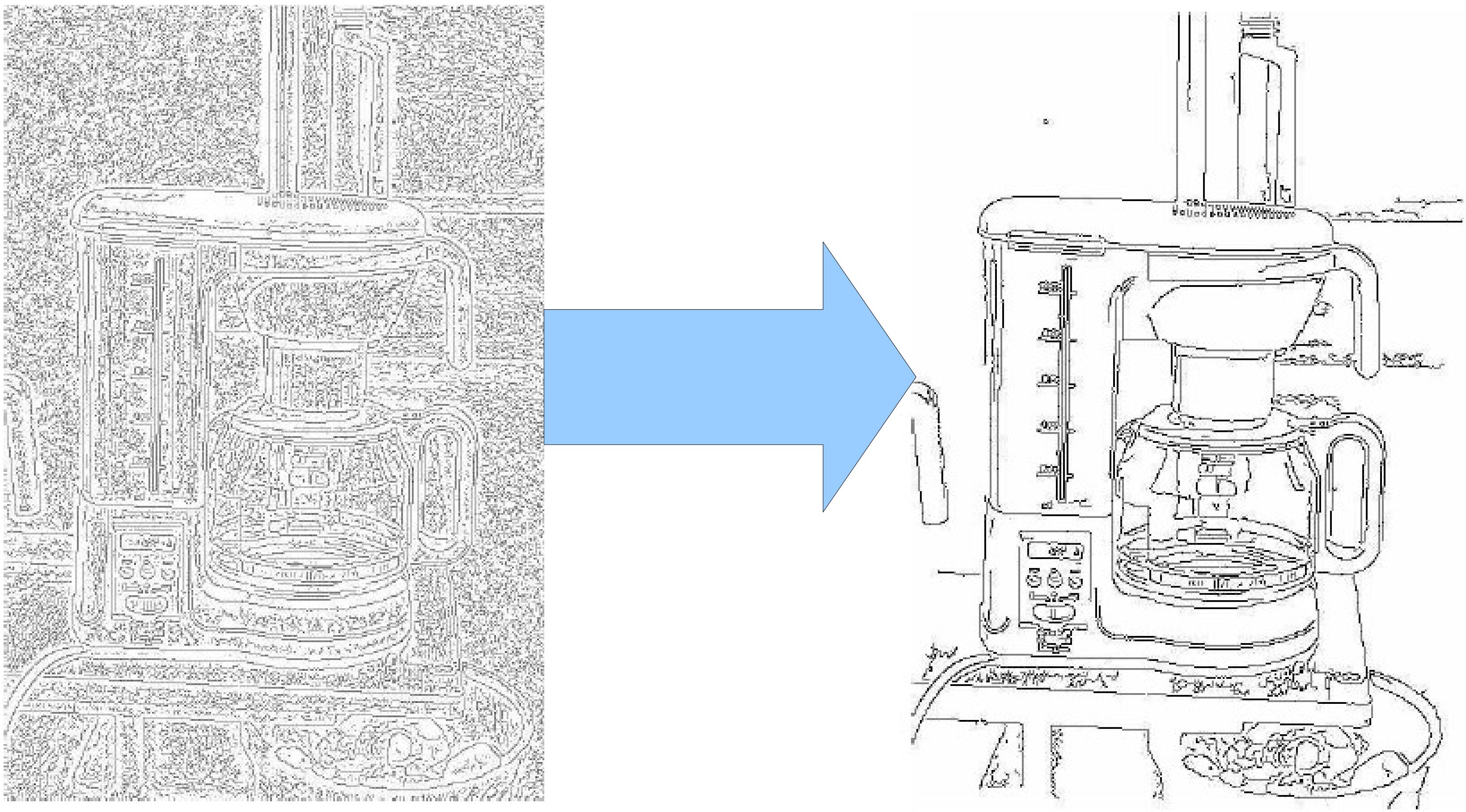
Single threshold value for edges -> Streaking

Two thresholds: low and high

- If a pixel value is above the high threshold, it is an edge.

- If a pixel value is below the low threshold, it is not an edge.

- If a pixel value is between the low and high thresholds,
it is an edge if it is connected to another edge pixel,
otherwise it is interpreted as noise.



Edge hysteresis

Homogeneous coordinates and duality point/line

homogenization

$$\mathbf{p} = [x\ \ y]^T \quad \Longrightarrow \quad \mathbf{p} = [x\ \ y\ \ 1]^T$$

Inhomogeneous vector — Homogeneous vector

dehomogenization (also known as Perspective division)

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \simeq \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}, \text{for } w \neq 0.$$

Projective plane $\mathbb{P}^2$

Equivalence class: $$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda w \end{bmatrix}, \forall \lambda \neq 0.$$

$L : ax + by + c = 0.$ is equivalent to $L : \lambda ax + \lambda by + \lambda c = 0$

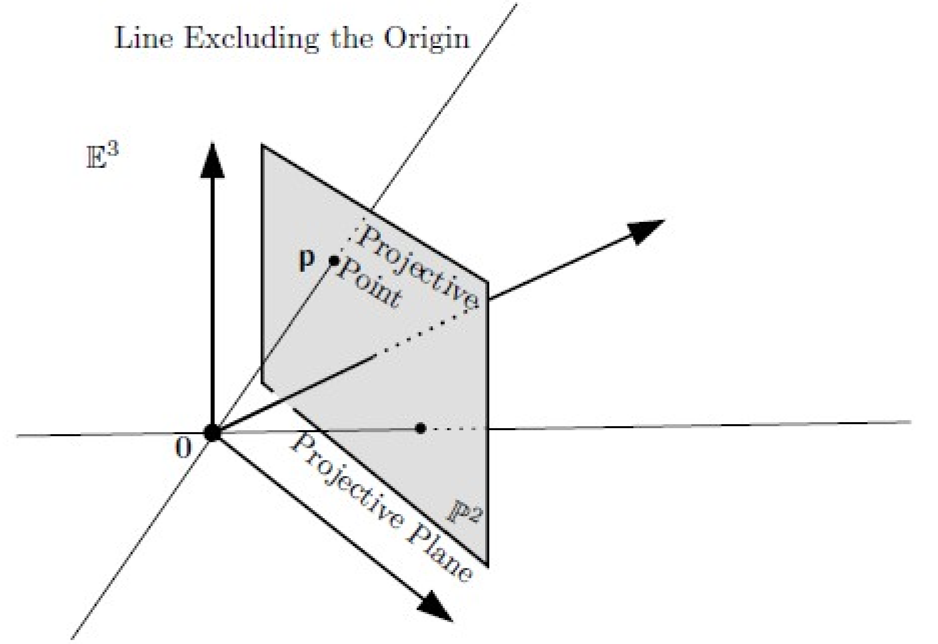
Line coefficients stored in an inhomogeneous vector $\mathbf{l} = \begin{bmatrix} a & b & c \end{bmatrix}^T$

Equation of the line: $L : \mathbf{l}^T \mathbf{p} = 0.$

Point and line have same homogeneous representation: A point can be interpreted as the coefficients of the line

Intersection of lines

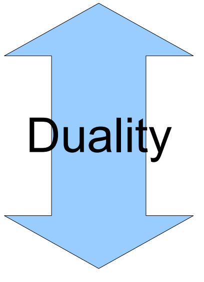
Cross-product of two vectors:



$$\mathbf{u} \times \mathbf{v} = \det \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{bmatrix} = -\mathbf{v} \times \mathbf{u}.$$

Intersection point of two lines is obtained from their cross-product:

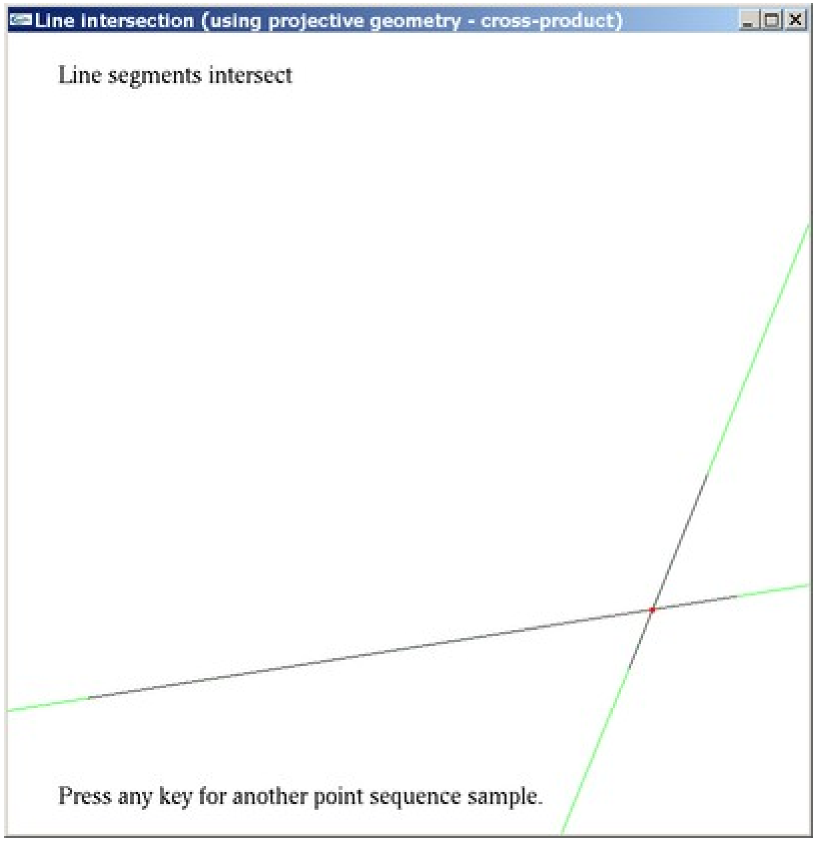
p= l1 x l2



Line passing through two « points » l1* and l2*:

l=p*=l1* x l2*

Application:
Detection of line segment intersection




```
l1=CrossProduct(p,q);
l2=CrossProduct(r,s);
// intersection point is the cross-product
//of the line coefficients (duality)
intersection=CrossProduct(l1,l2);
intersection.Normalize(); // to get back Euclidean point
```

Overview of duality in projective geometry

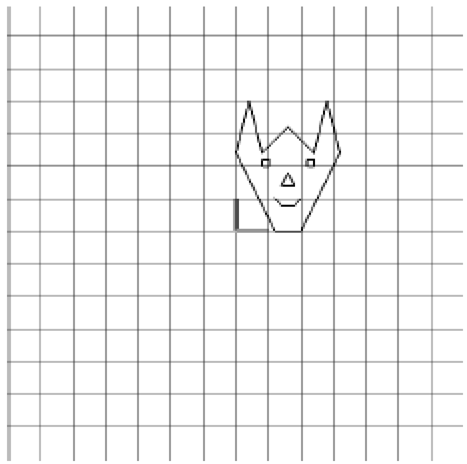
	Point	Line
Representation	$\mathbf{p} = \begin{bmatrix} x & y & w \end{bmatrix}^T$	$\mathbf{l} = \begin{bmatrix} a & b & c \end{bmatrix}^T$
Incidence	$\mathbf{p}^T \mathbf{l} = 0$ (lines $\mathbf{l}$ passing through $\mathbf{p}$)	$\mathbf{l}^T \mathbf{p} = 0$ (points $\mathbf{p}$ on line $\mathbf{l}$)
Degeneracy	Collinearity: $\det[\mathbf{p}_1 \ \mathbf{p}_2 \ \mathbf{p}_3] = 0$	Concurrence: $\det[\mathbf{l}_1 \ \mathbf{l}_2 \ \mathbf{l}_3] = 0$
Join	$\mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$ (line passing through $\mathbf{p}_1$ and $\mathbf{p}_2$)	$\mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2$ (intersection point of $\mathbf{l}_1$ and $\mathbf{l}_2$)
Infinity	Ideal points: $\begin{bmatrix} x & y & 0 \end{bmatrix}^T$	Ideal line: $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$

The determinant of three points represent the volume of their parallepiped. $(\mathbf{p}_1 \times \mathbf{p}_2) \cdot \mathbf{p}_3.$

2D Transformations using homogeneous coordinates

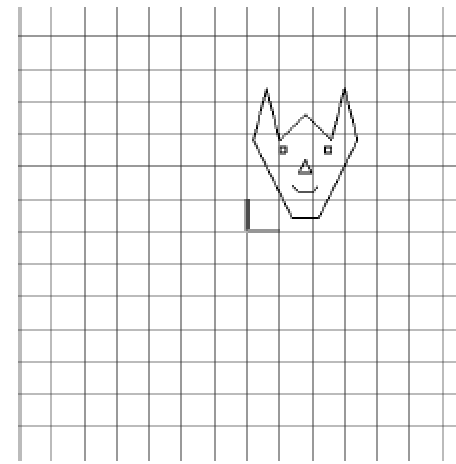
Identity **I**:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



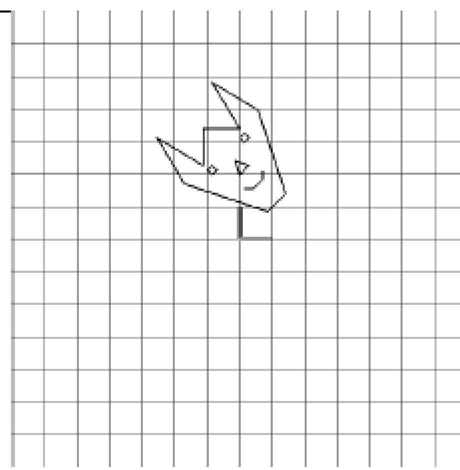
Translation **T**:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$



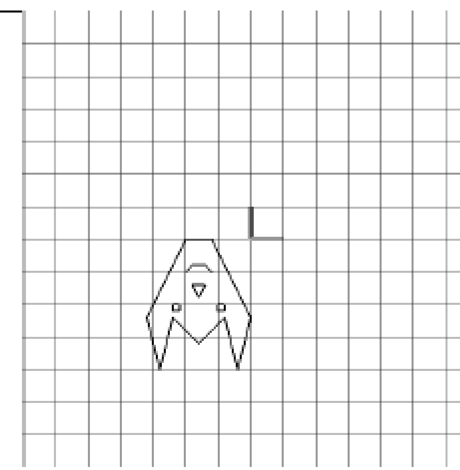
Rotation **R**:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 1 \\ \sin\theta & \cos\theta & 1 \\ 0 & 0 & 1 \end{bmatrix}$$



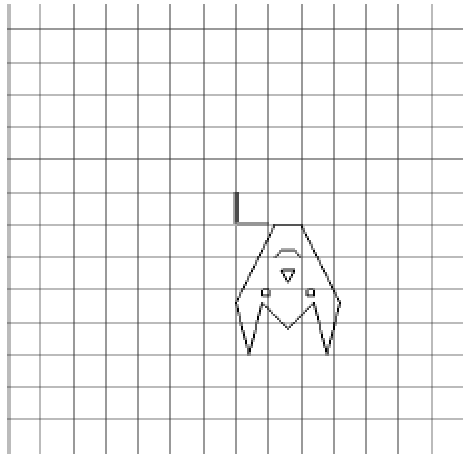
Central Symmetry **C** :

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

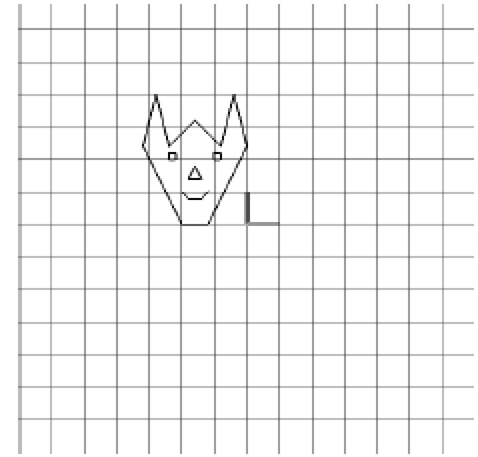


2D Transformations using homogeneous coordinates

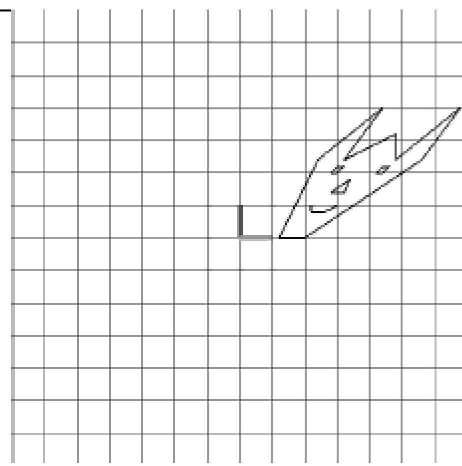
Y Symmetry $\mathbf{F_y}$:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



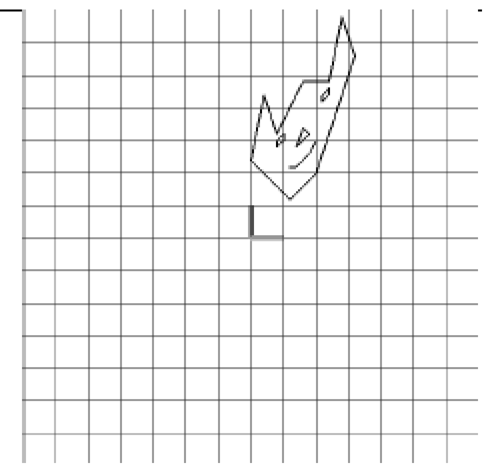
X Symmetry $\mathbf{F_x}$:
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



X Shear $\mathbf{S_{xy}}$
$(s = 1)$:
$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Y Shear $\mathbf{S_{yx}}$
$(s = 1)$:
$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Cartesian coordinate systems in 3D

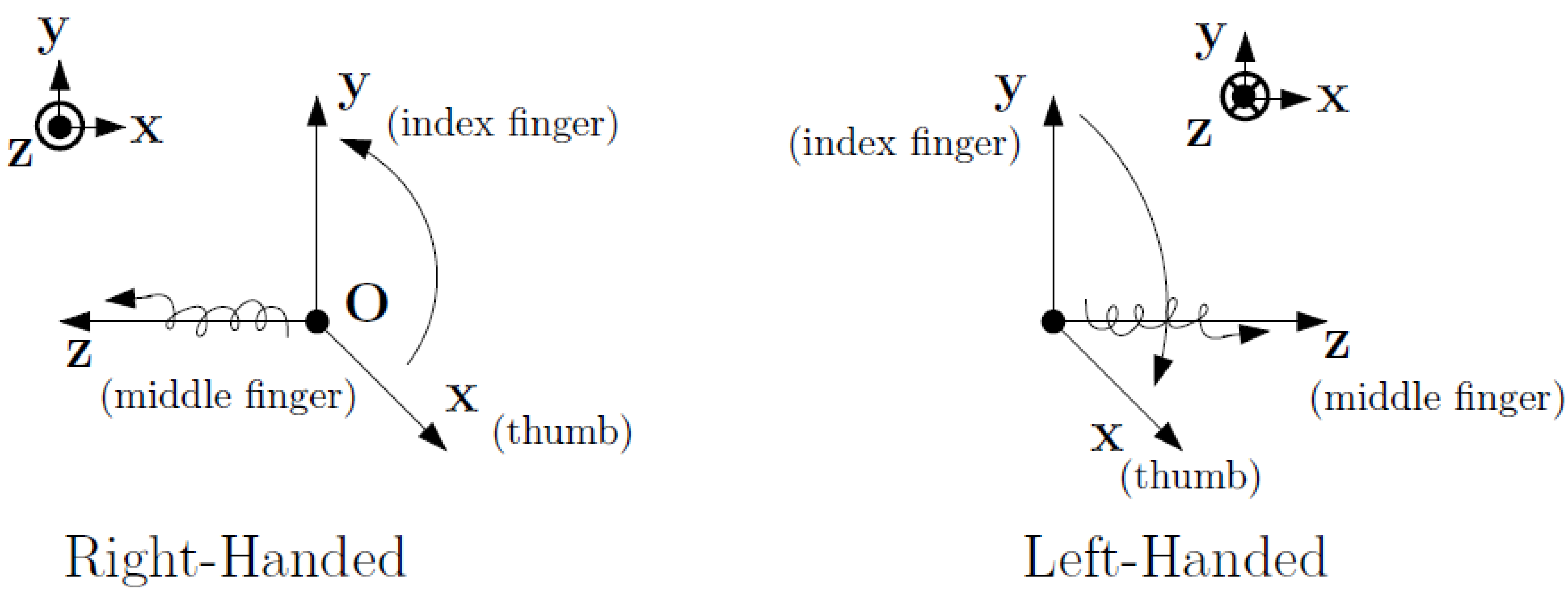


FIGURE 3.15 *The right-handed* ($\mathbf{z} = \mathbf{x} \times \mathbf{y}$) *and left-handed* ($\mathbf{z} = \mathbf{y} \times \mathbf{x} = -\mathbf{x} \times \mathbf{y}$) *Cartesian coordinate systems.*

3D Transformations using homogeneous coordinates

$$\mathsf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

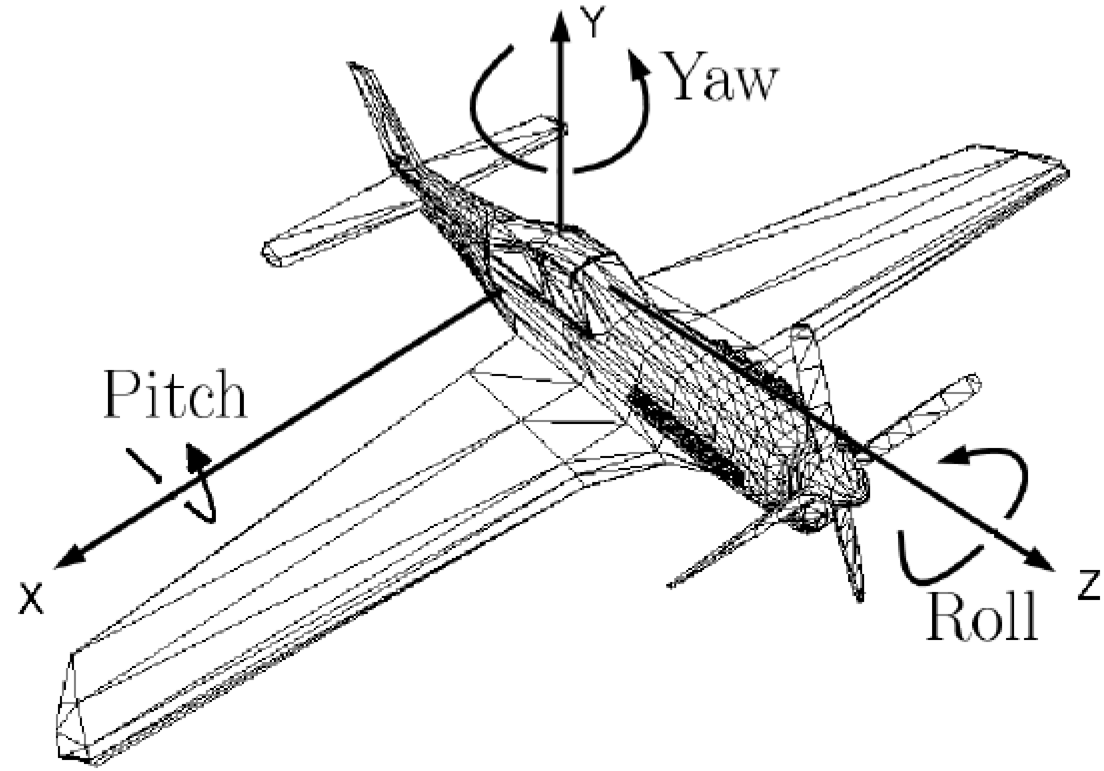
$$\mathsf{R}_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathsf{R}_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\mathsf{R} = \mathsf{R}_x \mathsf{R}_y \mathsf{R}_z$$

Be careful: Gimbal lock

Euler rotation



$$\mathbf{R}(\text{roll}, \text{pitch}, \text{yaw}) = \mathbf{R}_z(\text{roll}) \times \mathbf{R}_x(\text{pitch}) \times \mathbf{R}_{\boldsymbol{y}}(\text{yaw})$$

$$\mathbf{R}(\text{roll}, \text{pitch}, \text{yaw}) = \mathbf{R}(r, p, y) =$$
$$\begin{bmatrix} \cos r \cos y - \sin r \sin p \sin y & -\sin r \cos p & \cos r \sin y + \sin r \sin p \cos y \\ \sin r \cos y + \cos r \sin p \sin y & \cos r \cos p & \sin r \sin y - \cos r \sin p \cos y \\ -\cos p \sin y & \sin p & \cos p \cos y \end{bmatrix}$$

Cross-product/outer product

$$\mathbf{u} \times \mathbf{v} = \det \begin{bmatrix} \mathbf{x} & \mathbf{y} & \mathbf{z} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{bmatrix} = -\mathbf{v} \times \mathbf{u}.$$

Consider the cross-product as a matrix multiplication:

$$\mathbf{u} \times \mathbf{v} = [\mathbf{u}]_\times \mathbf{v}$$

$$[\mathbf{u}]_\times = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} = \mathbf{M}.$$

Outer-product

$$\mathbf{u}\mathbf{u}^T = \underbrace{\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}}_{(3,1)} \underbrace{\begin{bmatrix} u_x & u_y & u_z \end{bmatrix}}_{(1,3)} = \underbrace{\begin{bmatrix} u_x^2 & u_x u_y & u_x u_z \\ u_x u_y & u_y^2 & u_y u_z \\ u_x u_z & u_y u_z & u_z^2 \end{bmatrix}}_{(3,3)}$$

Arbitrary matrix rotation: Rodrigues' formula

$$\mathbf{R}_{\mathbf{u},\theta} = \mathbf{u}\mathbf{u}^T + \cos\theta(\mathbf{I} - \mathbf{u}\mathbf{u}^T) + [\mathbf{u}]_\times \sin\theta,$$

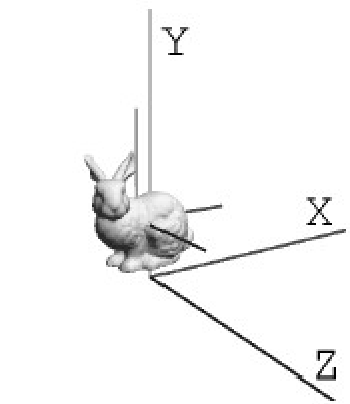
Equivalent to:

$$\mathbf{R}_{\mathbf{u},\theta} = \mathbf{I} + [\mathbf{u}]_\times \sin\theta + [\mathbf{u}]_\times^2 (1 - \cos\theta).$$

$$\mathbf{R}_{\mathbf{u},\theta} =
\begin{bmatrix}
\cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_y\sin\theta + u_x u_z(1-\cos\theta) \\
u_z\sin\theta + u_x u_y(1-\cos\theta) & \cos\theta + u_y^2(1-\cos\theta) & -u_x\sin\theta + u_y u_z(1-\cos\theta) \\
-u_y\sin\theta + u_x u_z(1-\cos\theta) & u_x\sin\theta + u_y u_z(1-\cos\theta) & \cos\theta + u_z^2(1-\cos\theta)
\end{bmatrix}.$$

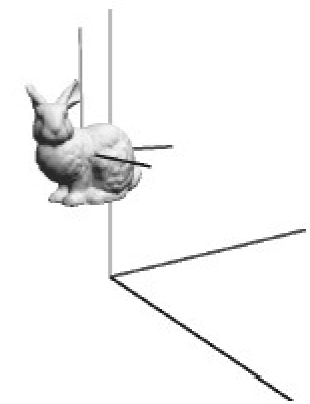
Identity $\mathbf{I}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



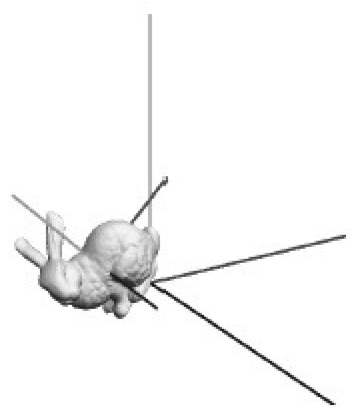
Translation $\mathbf{T}$:

$$\begin{bmatrix} 1 & 0 & 0 & -0.0268 \\ 0 & 1 & 0 & 0.095 \\ 0 & 0 & 1 & 0.009 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



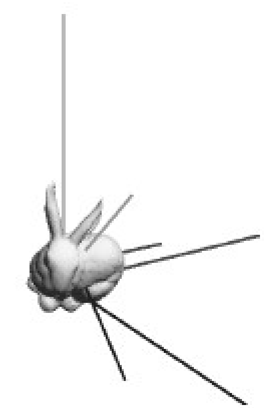
Rotation $\mathbf{R}_z$:
(45^o, z-axis)

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



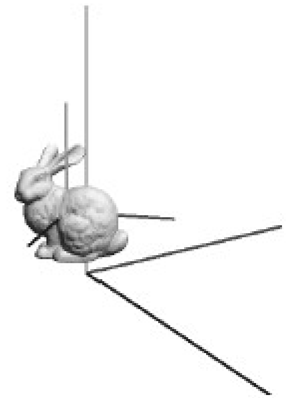
Rotation $\mathbf{R}_x$:
(45^o, x-axis)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



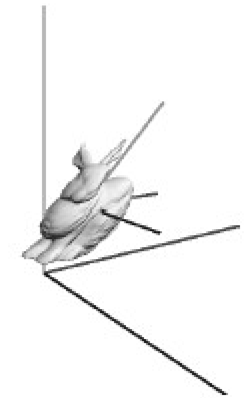
Rotation $\mathbf{R}_y$:
(45^o, y-axis)

$$\begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



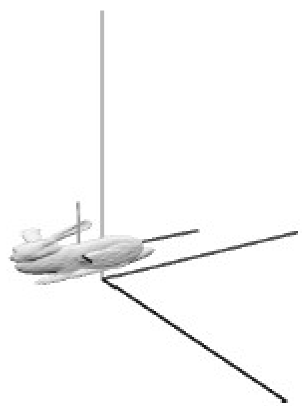
Shear $\mathbf{S_{xy}}$:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



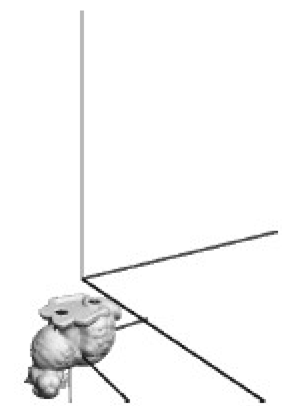
Scale $\mathbf{S}$:

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Y Symmetry $\mathbf{F_y}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Rigid transformations

$$\mathsf{D} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Concatenation (non-commutative!)

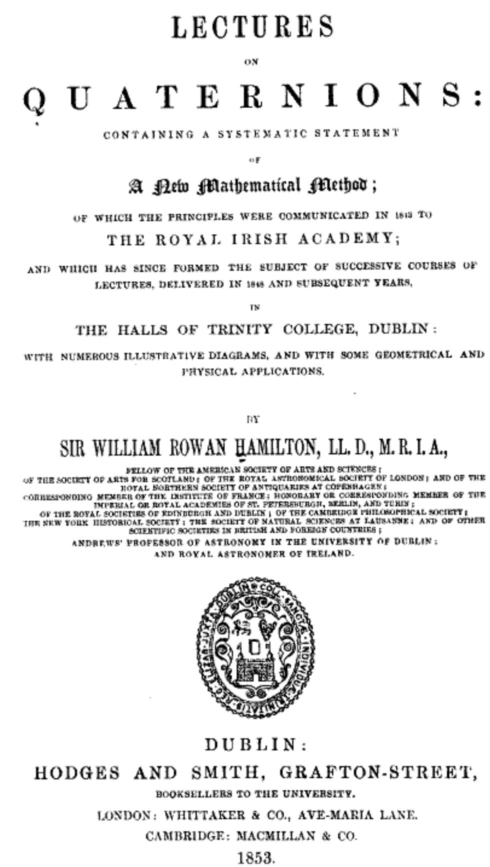
$$\mathsf{D}_1\mathsf{D}_2 = \begin{bmatrix} \mathrm{R}_1 & \mathrm{t}_1 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} \mathrm{R}_2 & \mathrm{t}_2 \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R}_1\mathrm{R}_2 & \mathrm{R}_1\mathrm{t}_2 + \mathrm{t}_1 \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R}' & \mathrm{t}' \\ 0^T & 1 \end{bmatrix} = \mathsf{D}'.$$

$$\mathsf{D}^{-1} = \begin{bmatrix} \mathrm{R}^T & -\mathrm{R}^T\mathrm{t} \\ 0^T & 1 \end{bmatrix}.$$

Quaternions for rotations

Provide rotation operator that is invertible

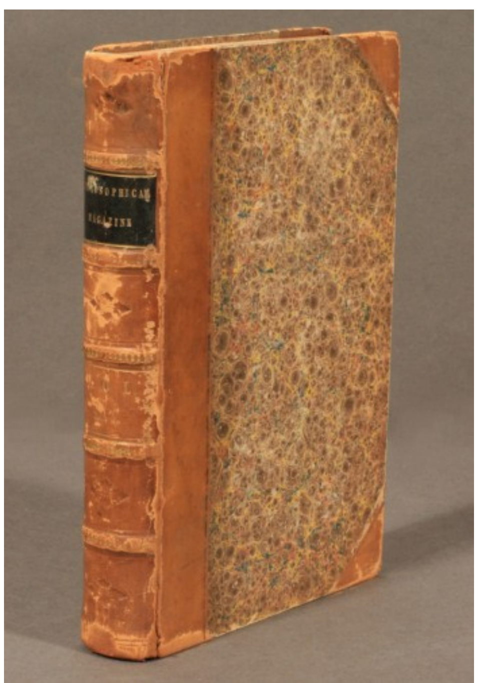
- Easy to invert scalars
- For 2D vectors, invert using complex numbers...
- For 3D vectors??? (-> 4D quaternions)
- For dD vectors??? (-> 8D octonions)


LECTURES
ON
QUATERNIONS:
CONTAINING A SYSTEMATIC STATEMENT
OF
A New Mathematical Method;
OF WHICH THE PRINCIPLES WERE COMMUNICATED IN 1843 TO
THE ROYAL IRISH ACADEMY;
AND WHICH HAS SINCE FORMED THE SUBJECT OF SUCCESSIVE COURSES OF LECTURES, DELIVERED IN 1848 AND SUBSEQUENT YEARS,
IN
THE HALLS OF TRINITY COLLEGE, DUBLIN:
WITH NUMEROUS ILLUSTRATIVE DIAGRAMS, AND WITH SOME GEOMETRICAL AND PHYSICAL APPLICATIONS.

BY

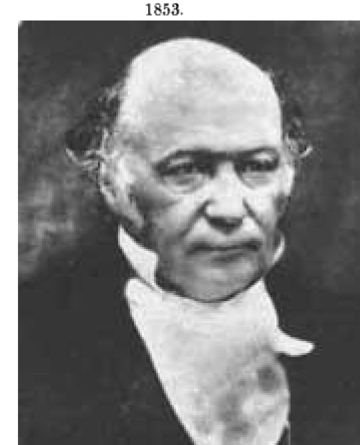
SIR WILLIAM ROWAN HAMILTON, LL. D., M. R. I. A.,
FELLOW OF THE AMERICAN SOCIETY OF ARTS AND SCIENCES;
OF THE SOCIETY OF ARTS FOR SCOTLAND; OF THE ROYAL ASTRONOMICAL SOCIETY OF LONDON; AND OF THE ROYAL NORTHERN SOCIETY OF ANTIQUARIES AT COPENHAGEN;
CORRESPONDING MEMBER OF THE INSTITUTE OF FRANCE; HONORARY OR CORRESPONDING MEMBER OF THE IMPERIAL OR ROYAL ACADEMIES OF ST. PETERSBURGH, BERLIN, AND TURIN;
OF THE ROYAL SOCIETIES OF EDINBURGH AND DUBLIN; OF THE CAMBRIDGE PHILOSOPHICAL SOCIETY;
THE NEW YORK HISTORICAL SOCIETY; THE SOCIETY OF NATURAL SCIENCES AT LAUSANNE; AND OF OTHER SCIENTIFIC SOCIETIES IN BRITISH AND FOREIGN COUNTRIES;
ANDREWS' PROFESSOR OF ASTRONOMY IN THE UNIVERSITY OF DUBLIN:
AND ROYAL ASTRONOMER OF IRELAND.

DUBLIN:
HODGES AND SMITH, GRAFTON-STREET,
BOOKSELLERS TO THE UNIVERSITY.
LONDON: WHITTAKER & CO., AVE-MARIA LANE.
CAMBRIDGE: MACMILLAN & CO.
1853.



Lectures on Quaternions

http://digital.library.cornell.edu/



Sir William Rowan Hamilton

Quaternions: 1D real+3D imaginary

$$\hat{\mathbf{q}} = [w \;\; \mathbf{u}]^T$$

Real part (1D)

Imaginary part (3D, i j k vectors)

Multiplication:

$$\hat{\mathbf{q}}_1 \hat{\mathbf{q}}_2 = \begin{bmatrix} w_1 w_2 - \mathbf{u_1} \cdot \mathbf{u_2} \\ \mathbf{u_1} \times \mathbf{u_2} + w_1 \mathbf{u_2} + w_2 \mathbf{u_1} \end{bmatrix}$$

Norm (l2)

$$||\hat{\mathbf{q}}|| = \sqrt{||\mathbf{u}||^2 + w^2}$$

Unit quaternions

$$\hat{\mathbf{q}} = \begin{bmatrix} \cos\theta \\ \mathbf{u}\sin\theta \end{bmatrix} \qquad ||\mathbf{u}|| = 1$$

Rotation theta around an axis u: Quaternion representation:

$$\hat{\mathbf{q}} = \begin{bmatrix} \cos\frac{\theta}{2} & \mathbf{u}\sin\frac{\theta}{2} \end{bmatrix}^T$$

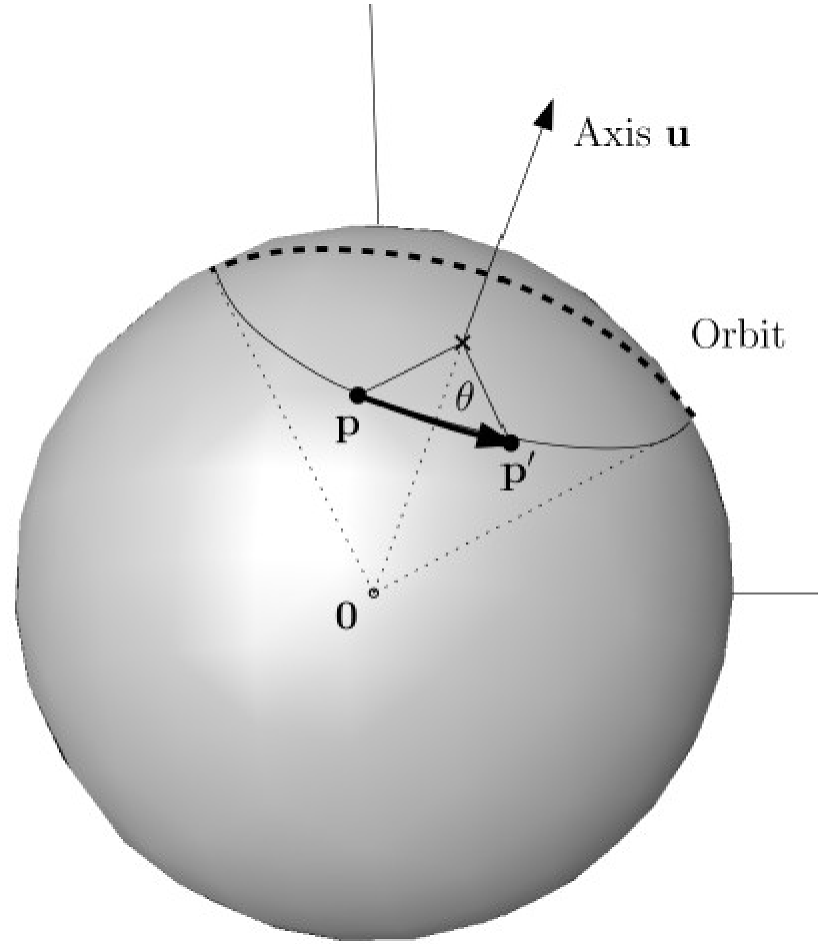
For a given 3D point p, we compute its rotation Rp as

$$\hat{\mathbf{p}}' = \hat{\mathbf{q}}[0\ \mathbf{p}]^T\hat{\mathbf{q}}^{-1} \qquad \hat{\mathbf{q}}^{-1} = \frac{\bar{\hat{q}}}{||\hat{\mathbf{q}}||}$$

$[w\ \ -\mathbf{u}]^T$

conjugate

Unit quaternions for rotations



$$\mathbf{p}' = \mathbf{R}\mathbf{p} \longleftrightarrow \hat{\mathbf{p}}' = \hat{\mathbf{q}}[0\ \mathbf{p}]^T\hat{\mathbf{q}}^{-1}$$

$$(\hat{\mathbf{q}} = [\cos\tfrac{\theta}{2}\ \ \mathbf{u}\sin\tfrac{\theta}{2}]^T)$$

$$\hat{\mathbf{q}} = [w\ \mathbf{u}]^T \longrightarrow \mathsf{R}(\hat{\mathbf{q}}) = \begin{bmatrix} 1-2u_y^2-2u_z^2 & 2u_xu_y-2wu_z & 2u_xu_z+2wu_y & 0 \\ 2u_xu_y+2wu_z & 1-2u_x^2-2u_z^2 & 2u_yu_z-2wu_x & 0 \\ 2u_xu_z-2wu_y & 2u_yu_z+2wu_x & 1-2u_x^2-2u_y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Conversion rotation matrix to quaternion

$$w = \frac{1}{2}\sqrt{\mathrm{trace}(\mathbf{R}) + 1}$$

$$\mathbf{u} = \begin{bmatrix} \frac{r_{yz} - r_{zy}}{4w} \\ \frac{r_{zx} - r_{xz}}{4w} \\ \frac{r_{xy} - r_{yz}}{4w} \end{bmatrix}$$

Spherical linear interpolation (SLERP)

LERP is non-sense for rotation matrices:

$$\mathbf{R}_\lambda = (1-\lambda)\mathbf{R}_0 + \lambda\mathbf{R}_1,$$

$$\mathbf{R}_\lambda = \mathbf{R}_0 + \lambda(\mathbf{R}_1 - \mathbf{R}_0) = \mathrm{LERP}(\mathbf{R}_0, \mathbf{R}_1; \lambda).$$

SLERP is using quaternion algebra:

$$\hat{\mathbf{q}}_\lambda = (\hat{\mathbf{q}}_2\hat{\mathbf{q}}_1^{-1})^\lambda \hat{\mathbf{q}}_1$$

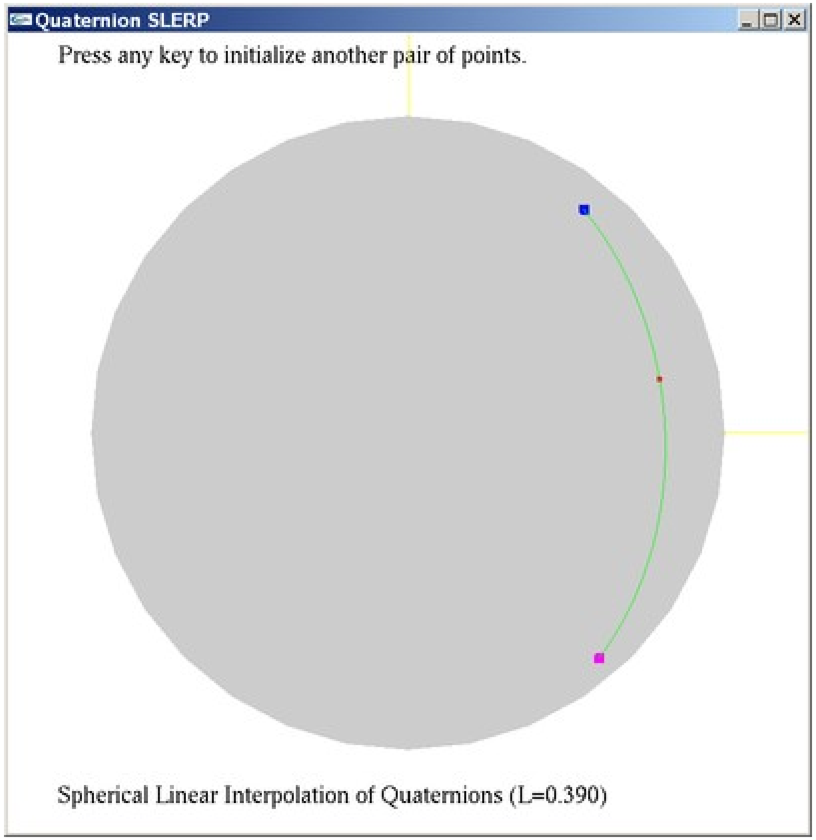
$$\hat{\mathbf{q}}^\lambda = (\exp(\theta\mathbf{u}))^\lambda = \exp(\lambda\theta\mathbf{u}) = \cos\lambda\theta + (\sin\lambda\theta)\mathbf{u}.$$

$$\mathrm{SLERP}(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2; \lambda) = \frac{\hat{\mathbf{q}}_1 \sin(1-\lambda)\theta + \hat{\mathbf{q}}_2 \sin\lambda\theta}{\sin\theta}$$

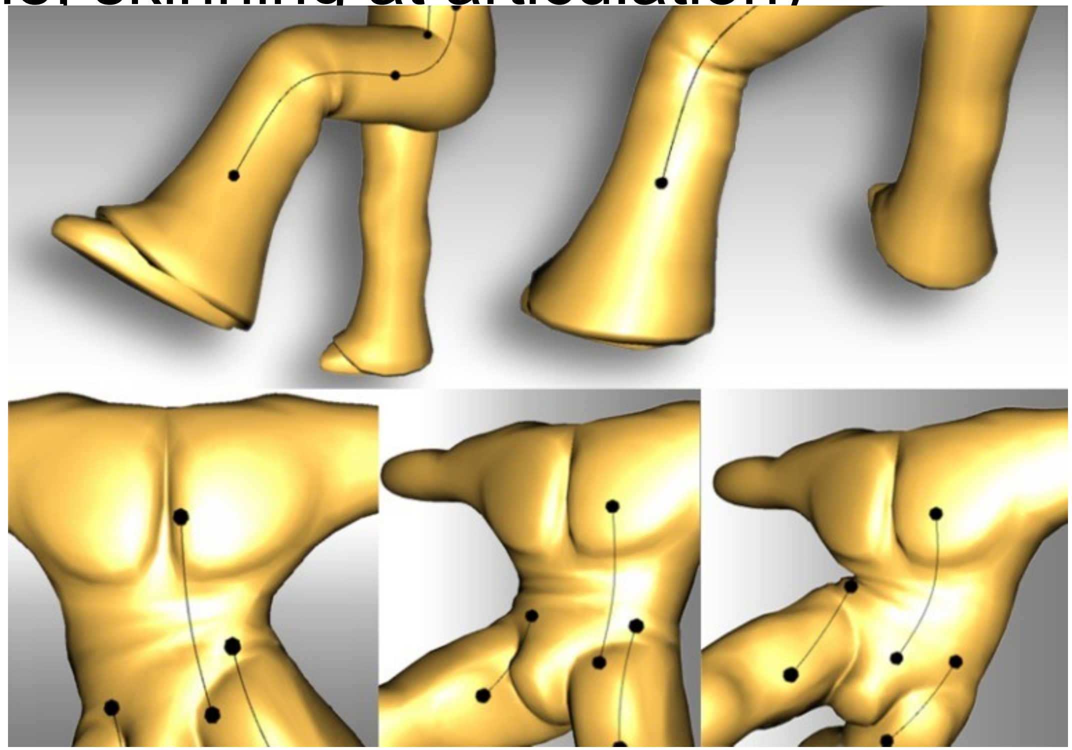
$$\mathrm{SLERP}(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2; \lambda) \simeq_{\theta\to 0} (1-\lambda)\hat{\mathbf{q}}_1 + \lambda\hat{\mathbf{q}}_2 = \mathrm{LERP}(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2; \lambda)$$

Spherical linear interpolation (SLERP)

$$\mathrm{SLERP}(\hat{\mathrm{q}}_1, \hat{\mathrm{q}}_2; \lambda) = \frac{\hat{\mathrm{q}}_1 \sin(1-\lambda)\theta + \hat{\mathrm{q}}_2 \sin \lambda\theta}{\sin \theta}$$

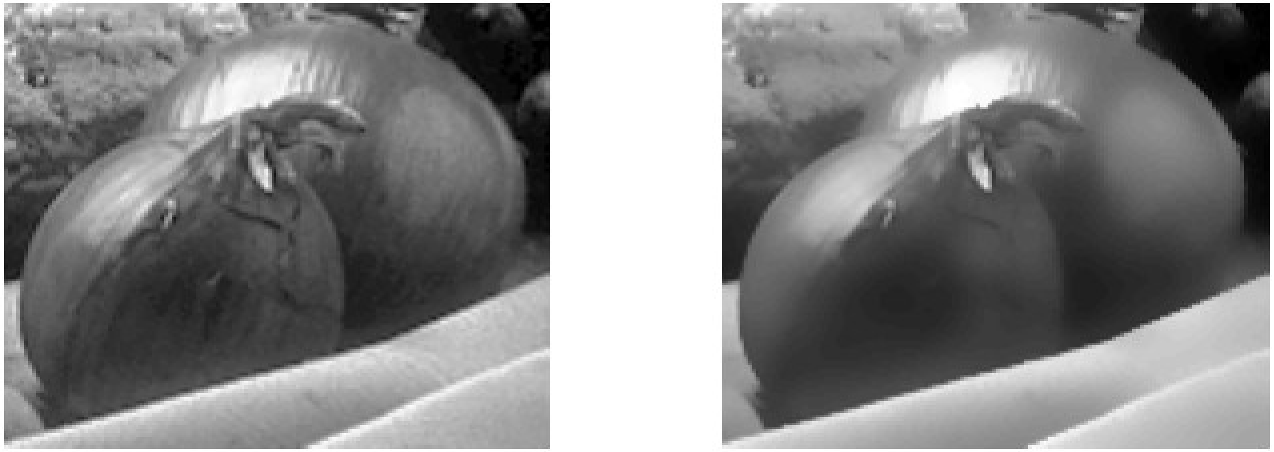


Useful for computer graphics animation (bone, skinning at articulation)



Bilateral filtering

Edge-preserving smoothing

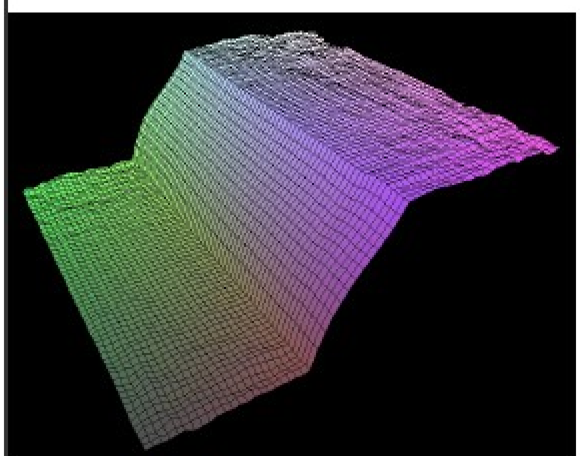


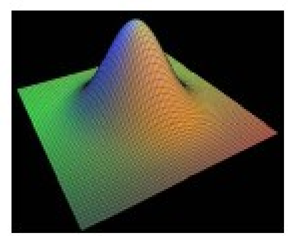
Videos/demo

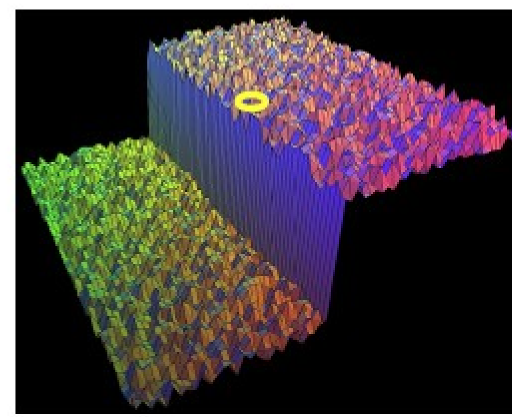
Gaussian filtering: Blur everything

Traditional spatial gaussian filtering

$$J(x) = \sum_{\xi} f(x,\xi) \, I(\xi)$$





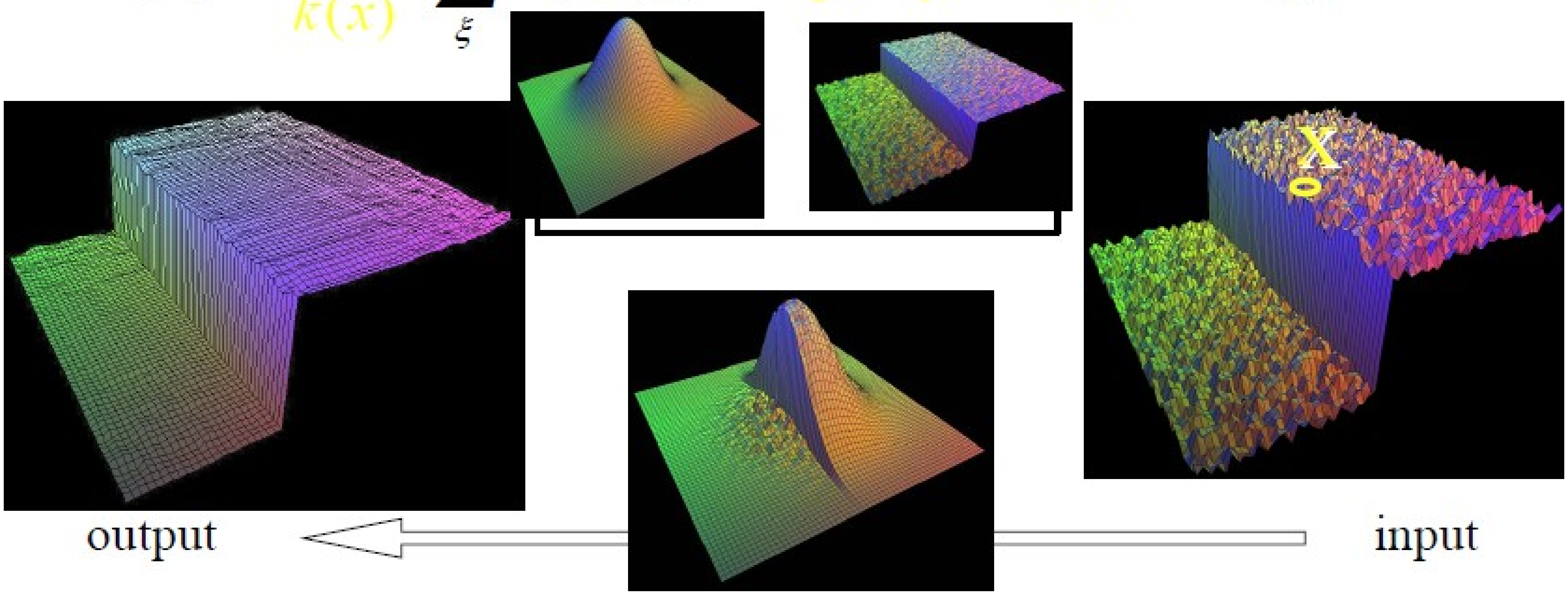


output ⟵ input

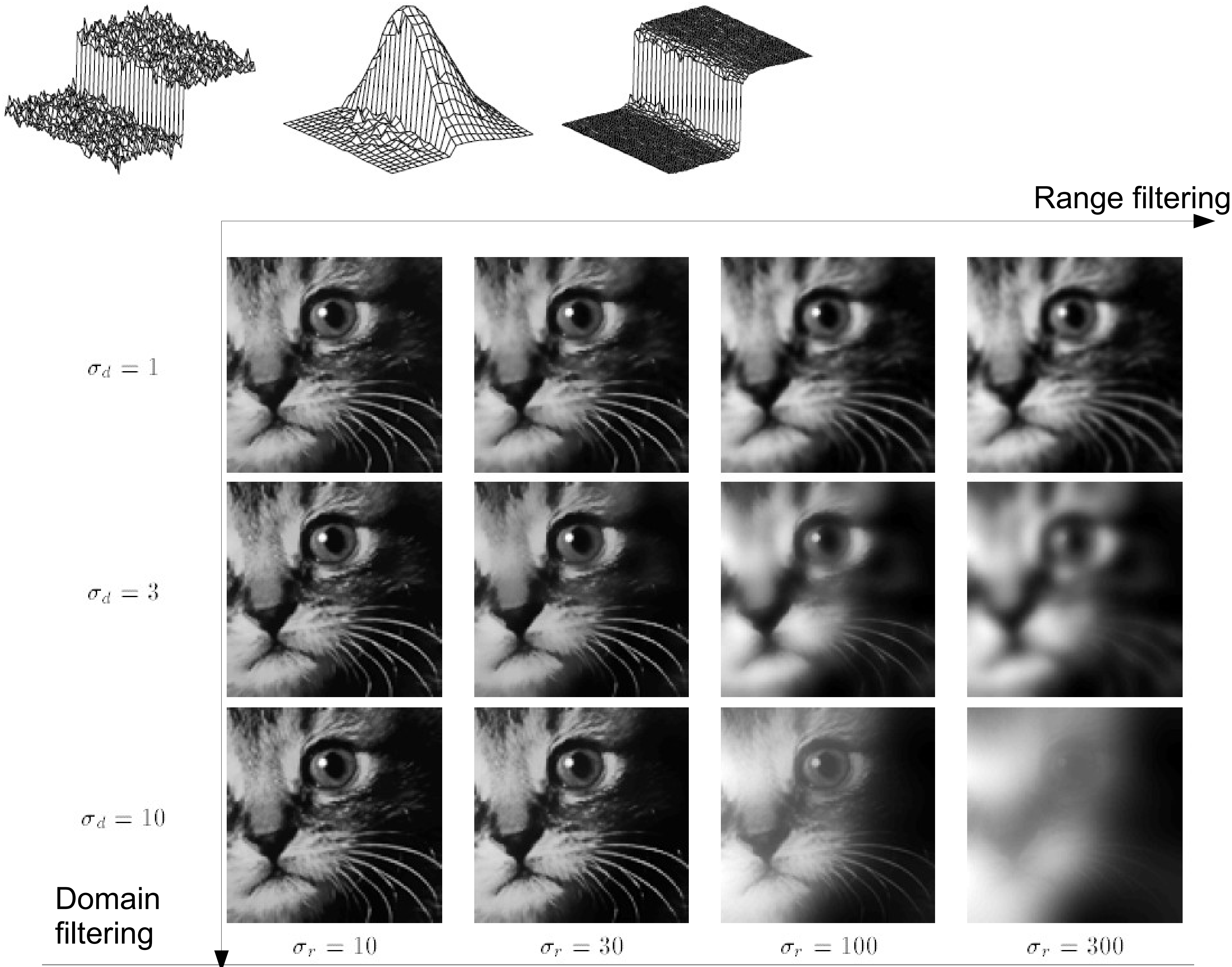
Bilateral filtering

New! gaussian on the intensity difference filtering

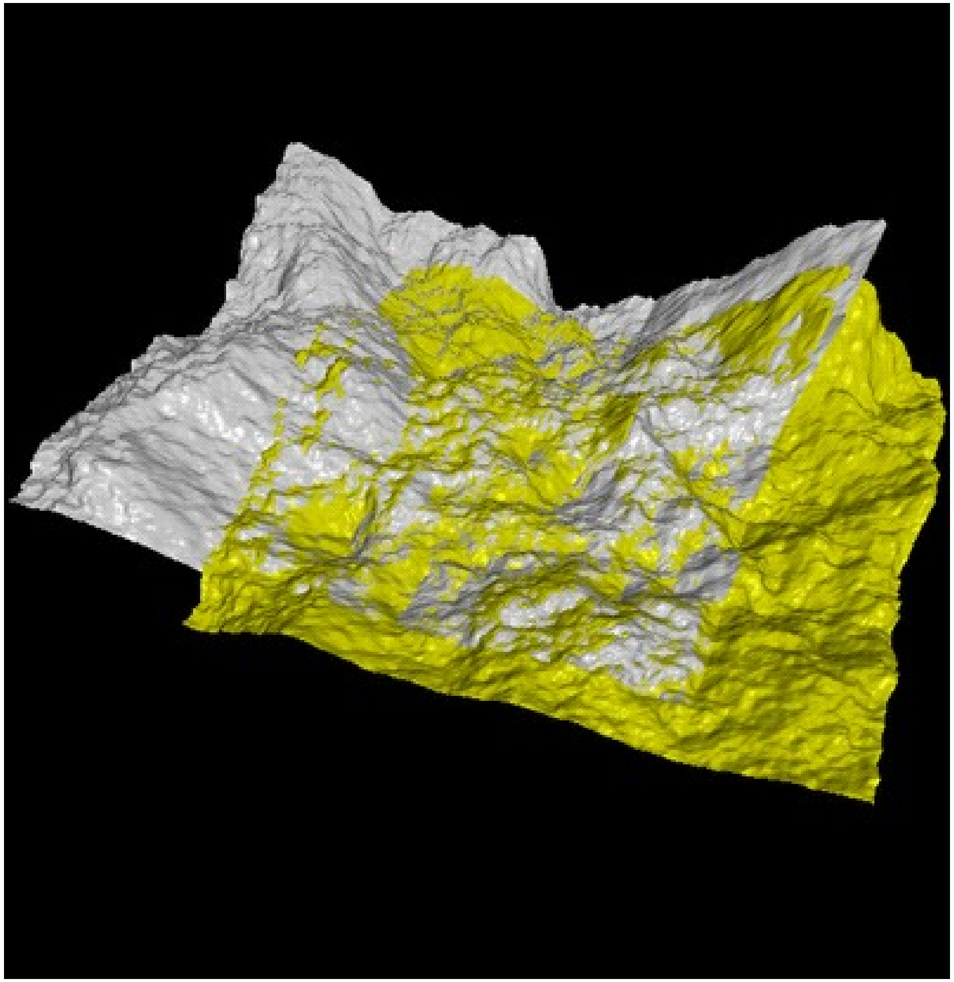
$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi) \quad g(I(\xi)-I(x)) \quad I(\xi)$$

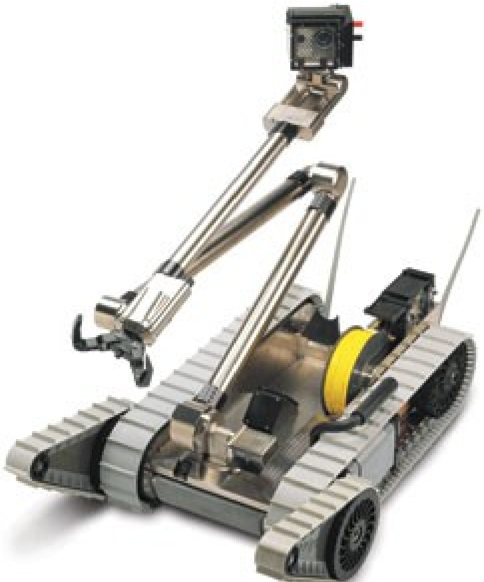


Bilateral Filtering for Gray and Color Images, Tomasi and Manduchi 1998
.... SUSAN feature extractor...


Range filtering
$\sigma_d = 1$
$\sigma_d = 3$
$\sigma_d = 10$
Domain filtering
$\sigma_r = 10$
$\sigma_r = 30$
$\sigma_r = 100$
$\sigma_r = 300$

Iterative Closest Point (ICP)

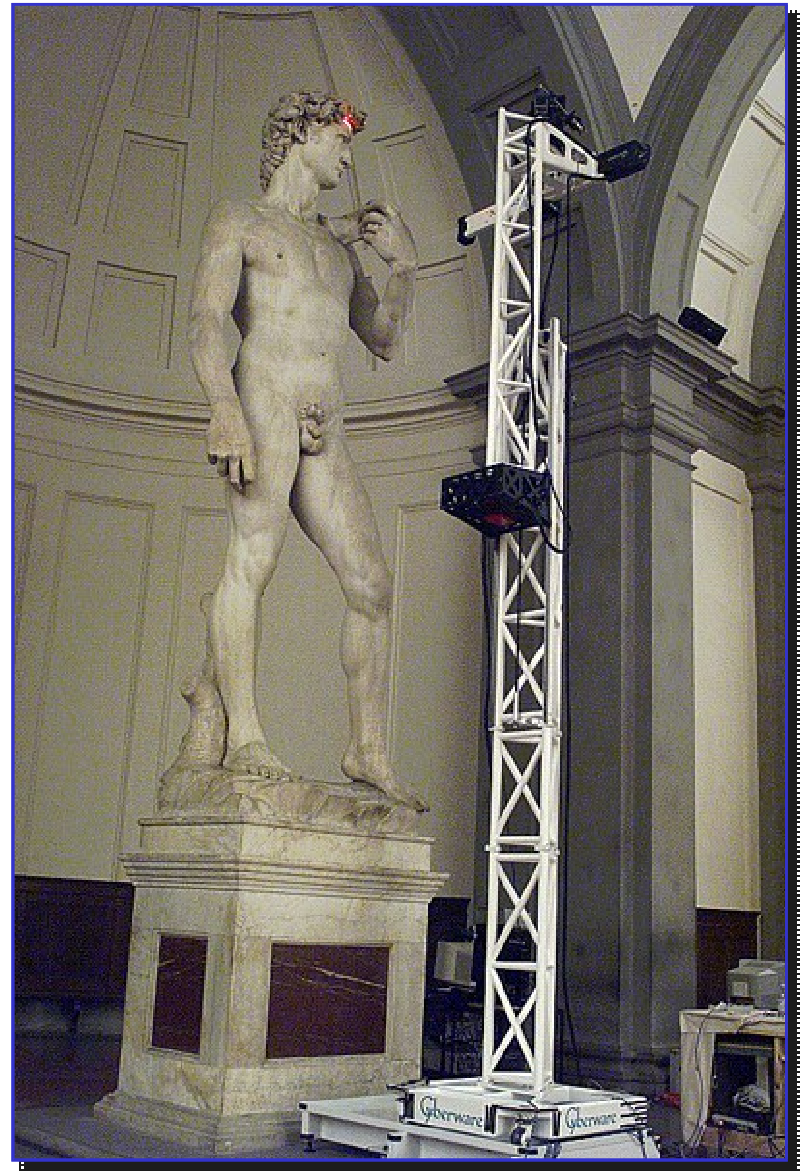


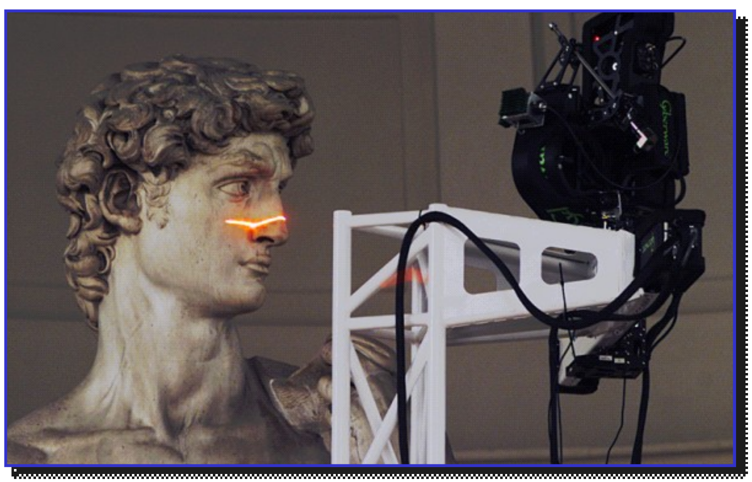


Robotics

Align point sets. For example, terrains (DEMs)

Align point sets obtained from range scanners





http://www-graphics.stanford.edu/projects/mich/

ICP for solving jigsaws

Solve stone jigsaws...

ICP: Algorithm at a glance

- Start from a not too far transformation

- Match the point of the target to the source
- Compute the best transformation from point correspondence
- Reiterate until the mismatch error goes below a threshold

In practice, this is a very fast registration method...

A Method for Registration of 3-D Shapes. by: Paul J *Besl*, Neil D M
IEEE Trans. Pattern Anal. Mach. Intell., Vol. 14, No. 2. (February 19

ICP: Finding the best rigid transformation

Given point correspondences, find the best rigid transformation.

$$X = \{x_1, ..., x_n\}$$

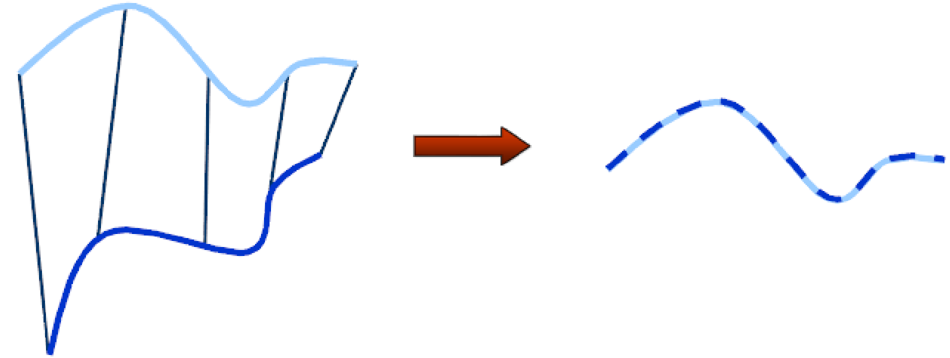
Observation/Target

$$P = \{p_1, ..., p_n\}$$

Source/Model

Find (R,t) that minimizes the squared euclidean error:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2$$



Align the center of mass of sets:

$$\mu_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \quad \text{and} \quad \mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$$

$$\begin{aligned} X &= \{x_1, ..., x_n\} \\ P &= \{p_1, ..., p_n\} \end{aligned} \quad \Longrightarrow \quad \begin{aligned} X' &= \{x_i - \mu_x\} = \{x'_i\} \\ P' &= \{p_i - \mu_p\} = \{p'_i\} \end{aligned}$$

Finding the rotation matrix:

$$W = \sum_{i=1}^{N_p} x'_i p'^T_i$$

Compute the singular value decomposition

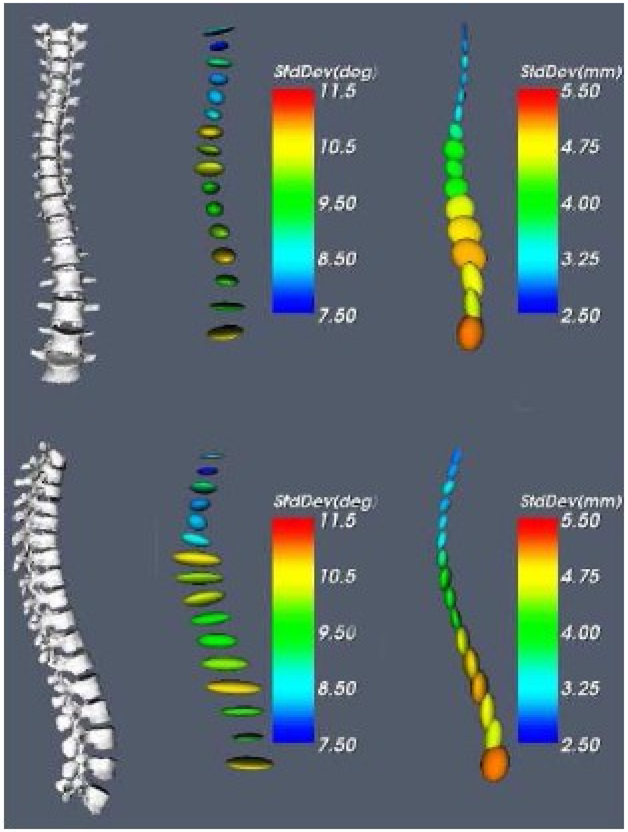
$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

$$\sigma_1 \geq \sigma_2 \geq \sigma_3$$

Optimal transformation:

$$R = UV^T$$
$$t = \mu_x - R\mu_p$$

Registration of many point sets to a common atlas



Scoliotic Spine (Atlas of 307 patients)

Many variants of ICP method (truncated, robust, etc.)

What computational geometers say

In theory,

ICP may *provably* run very slowly for **well-constructed** point sets...

David Arthur; Sergei Vassilvitskii

Worst-case and Smoothed Analysis of the ICP Algorithm, with an Application to the k-means Method

FOCS 2006 => O(n/d)^d iterations (exponential)

... but smooth analysis of ICP is polynomial

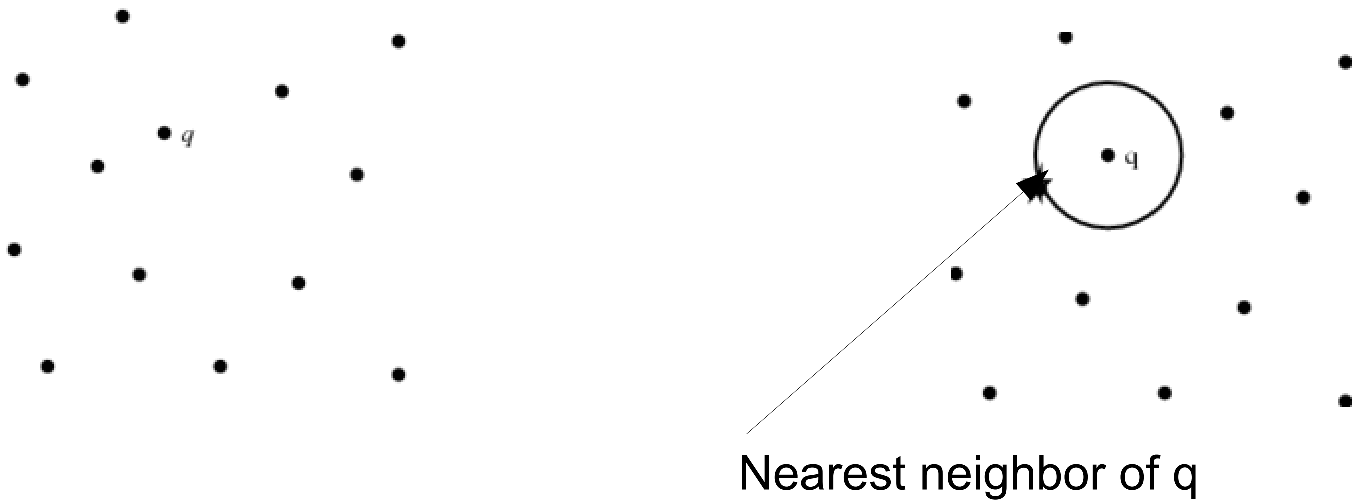
Theorem. With probability $1-2p$ ICP will finish after at most

$$O(n^{11}d\left(\frac{D}{\sigma}\right)^2 p^{-2/d})$$ iterations.

Since ICP always runs in at most $O(dn^2)^d$ iterations, we can take

$p = O(dn^2)^{-d}$ to show that the smoothed complexity is polynomial.

Computing nearest neighbors in ICP...



- Naive linear-time algorithm
- Tree-like algorithm using kd-trees
- Tree-like algorithm using metric ball trees
- ...

Challenging problem in very high-dimensions
(common to work up to dimension > 1000 nowadays)

Installez JOGL sur vos machines svp

Java OpenGL https://jogl.dev.java.net/