Rotations for Computer Vision and Robotics

Umberto Castellani

Robotics, vision and control

Rotations

- Rotations, and spatial transformations, are very important for many task in computer vision and robotics.
- Rotations should be:
  - Composed,
  - Inverted,
  - Differentiated,
  - interpolated

Each of these aspects can be better addressed with a particular rotation techniques!

Rotations

- Rotation matrix,
- Euler angles,
- Axis-angles representation,
- Quaternions,
- Exponential matrix

Rotations as 3x3 matrices (9 scalars)

- after all, rotations are linear operators
- Rot = 3x3 submatrix of a 4x4 rotation affine matrix

$$\left[\begin{array}{ccc|c} & & & 0 \\ & R & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

- Reminder: R is orthonormal, with det = +1

Rotations as 3x3 matries (9 scalars)

- Wasteful in RAM (9 scalars, versus a minimum of 3)
- Easy to apply (matrix-vector prod: 9 mults)
- Relat. easy to compose (matrix-matrix prod: 27 x mult)
- Immediate to invert (just transpose)
- Interpolate: troubles

$$k\,\boxed{R_0} + (1-k)\,\boxed{R_1} = \boxed{M}$$

NOT a rotation

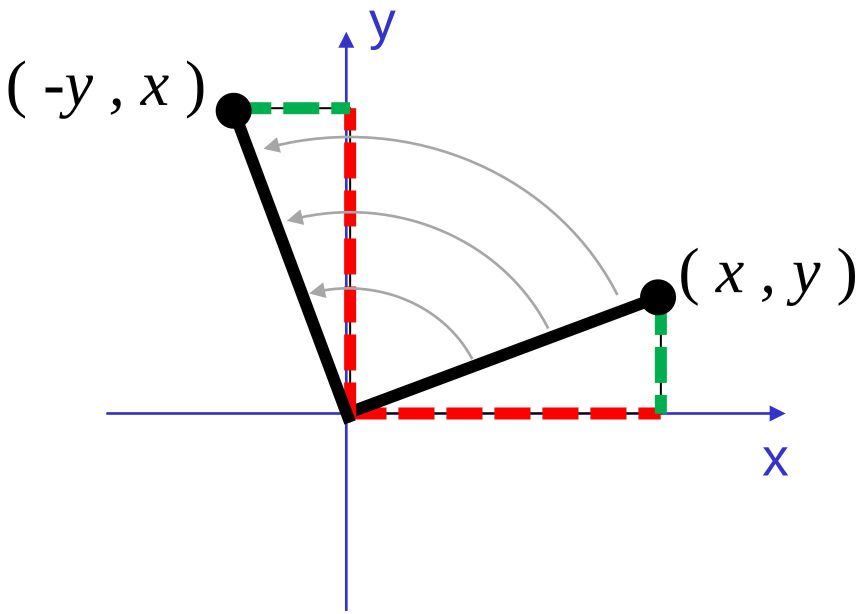
Compositions

- Multiplying matrices composites the rotation
  - remember: neither matrix-matrix product, nor composition of 3D rotations, is commutative!
- e.g.: $R_{TOT} = R_0 \cdot R_1$
  - rotate as R_1 followed by R_0
  - with $R_0 \cdot R_1$ rotation matrices
  - i.e. orthonormal matrices with det = 1
- R_{TOT} is a rotation matrix too, in theory
- in practice, approximation errors can break that
  - especially after long sequences of compositions.

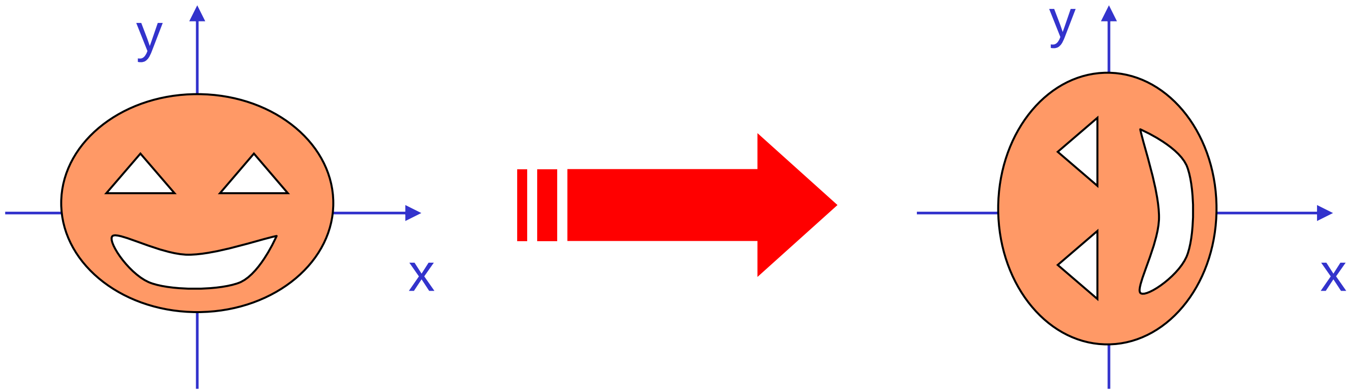
Rotations as 3x3 matrices (9 scalars)

- Nice plus:
  its three columns are
  the three **versors** representing
  the **X** , **Y** , **Z** axis of the *local* space
  in global space
  - i.e. the world-space versors

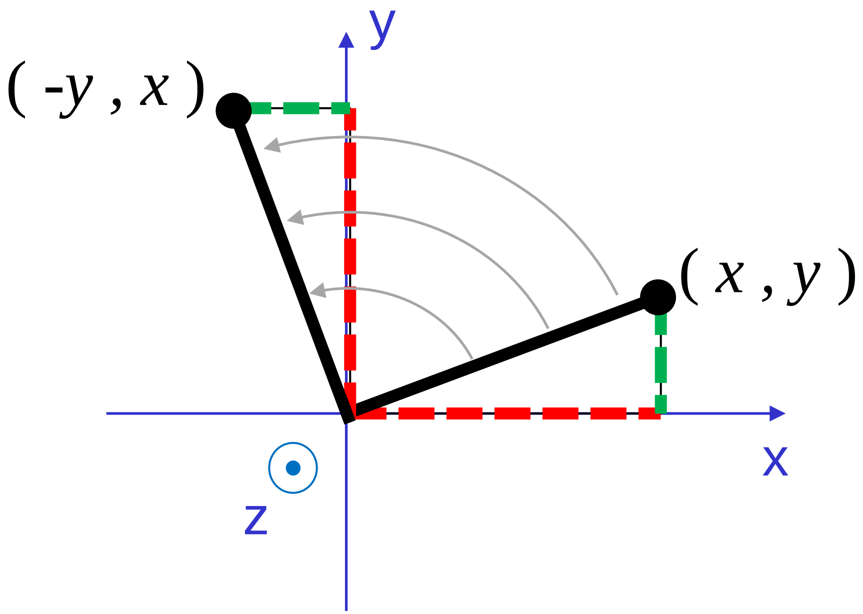
Osservazione:
trasf. di **rotazione di 90 gradi senso antiorario**



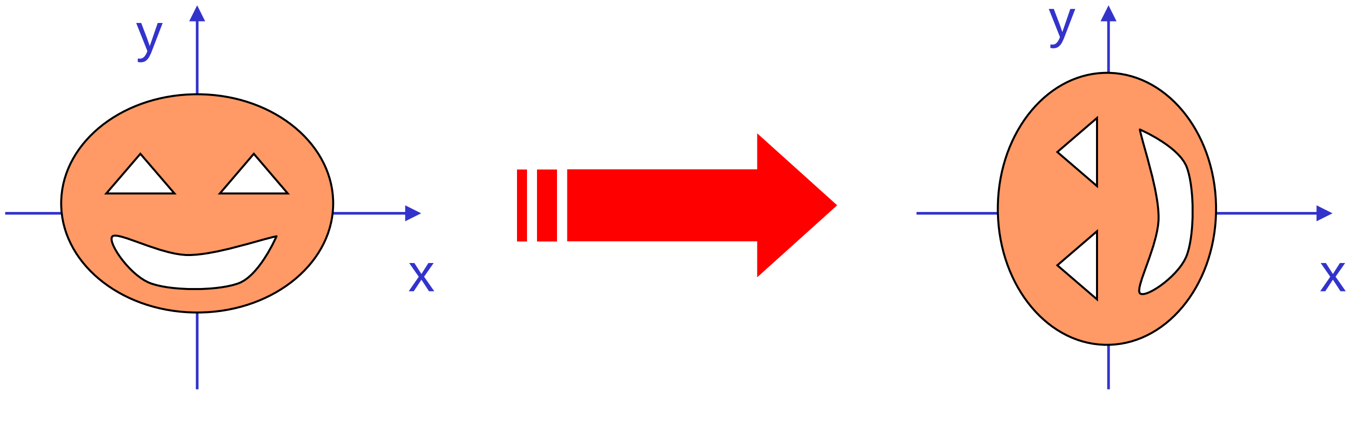
$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix}$$



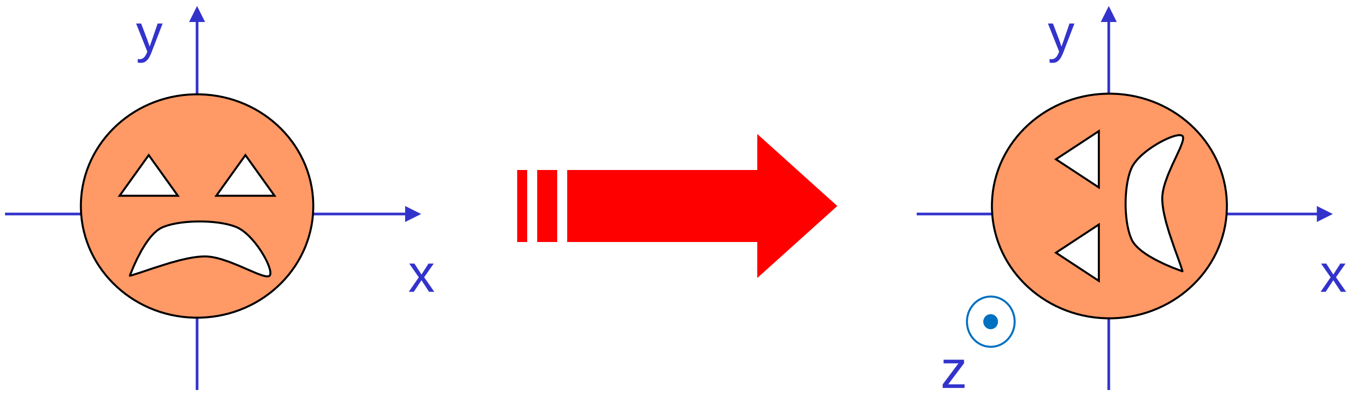
In 3D:
trasf. di **rotazione di 90 gradi attorno all'asse delle z**



$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -y \\ x \\ z \end{pmatrix}$$



Matrice di trasformazione: rotazione attorno all'asse delle Z



$$f\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} -y \\ x \\ z \\ 1 \end{pmatrix}$$

$$\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -y \\ x \\ z \\ 1 \end{bmatrix}$$

$$\mathbf{R}_{z,90°}$$

matrice di rotazione di 90° attorno all'asse delle Z

Rotazioni in 3D: esercizi

- Es: definire la trasf f
  - ⇒ di rotazione di 180° attorno all'asse delle Z
  - ⇒ di 90° attorno all'asse delle X e Y

Come cambia la matrice di trasformazione?

Trasf. di rotazione in 2D generica
(attorno all'origine, di un angolo β **generico**, in senso antiorario)

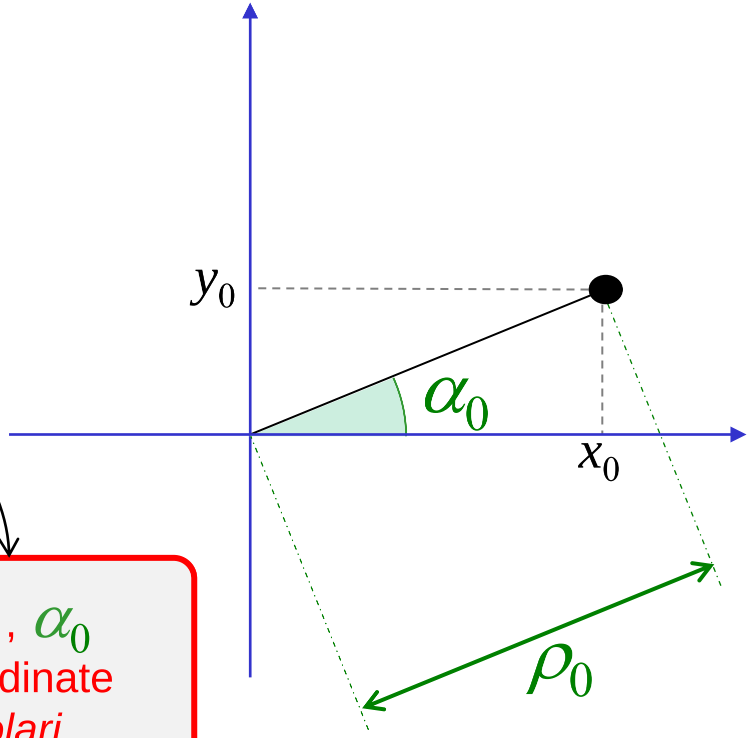
preliminare:coordinate polari
(un modo diverso per
esprimere punti / vettori in 2D)

$$\rho_0 = \sqrt{x_0^{\,2} + y_0^{\,2}}$$
$$\alpha_0 = atan2(\, y_0 \,,\, x_0 \,)$$

$x_0 \,,\, y_0$
coordinate
catersiane

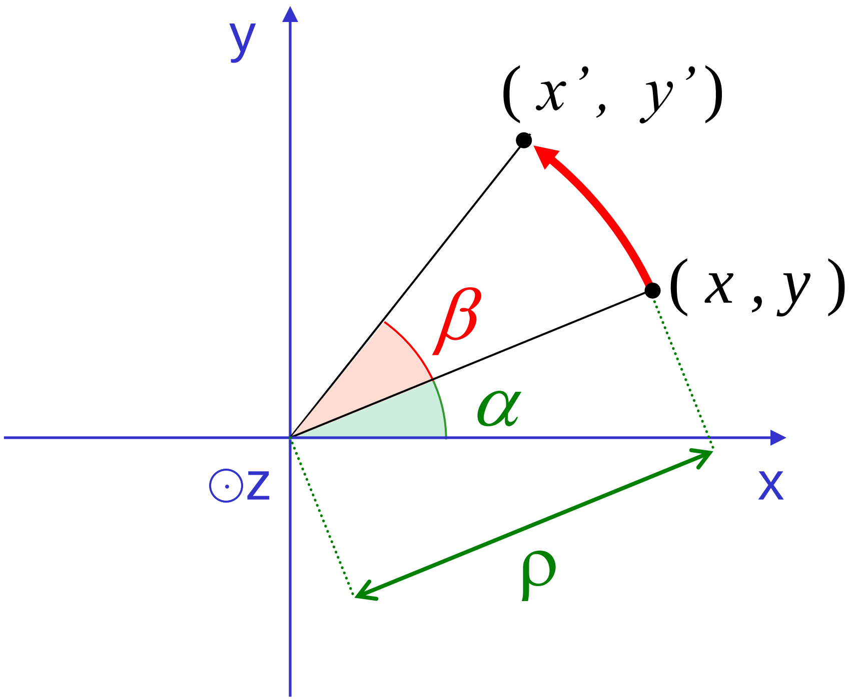
$\rho_0 \,,\, \alpha_0$
coordinate
polari

$$x_0 = \rho_0 \cdot \cos(\alpha_0)$$
$$y_0 = \rho_0 \cdot \sin(\alpha_0)$$



Trasf. di rotazione in 2D
(attorno all'origine, di un angolo β **generico**, in senso antiorario)

In coordinate polari,
la rotazione sarebbe banale:
l'angolo α incrementa di β,
la distanza ρ rimane invariata



partenza:

$$x = \rho \cos \alpha$$

$$y = \rho \sin \alpha$$

arrivo:

$$x' = \rho \cos(\alpha + \beta) = \rho \cos \alpha \cos \beta - \rho \sin \alpha \sin \beta = x \cos \beta - y \sin \beta$$

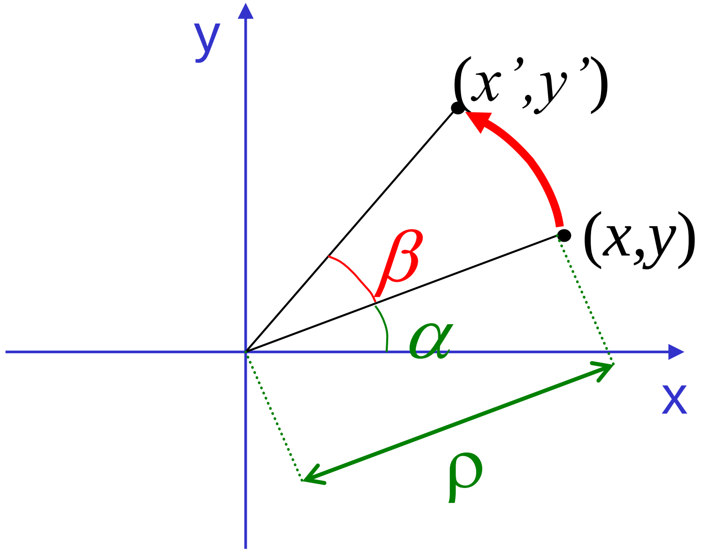
$$y' = \rho \sin(\alpha + \beta) = \rho \cos \alpha \sin \beta + \rho \sin \alpha \cos \beta = x \sin \beta + y \cos \beta$$

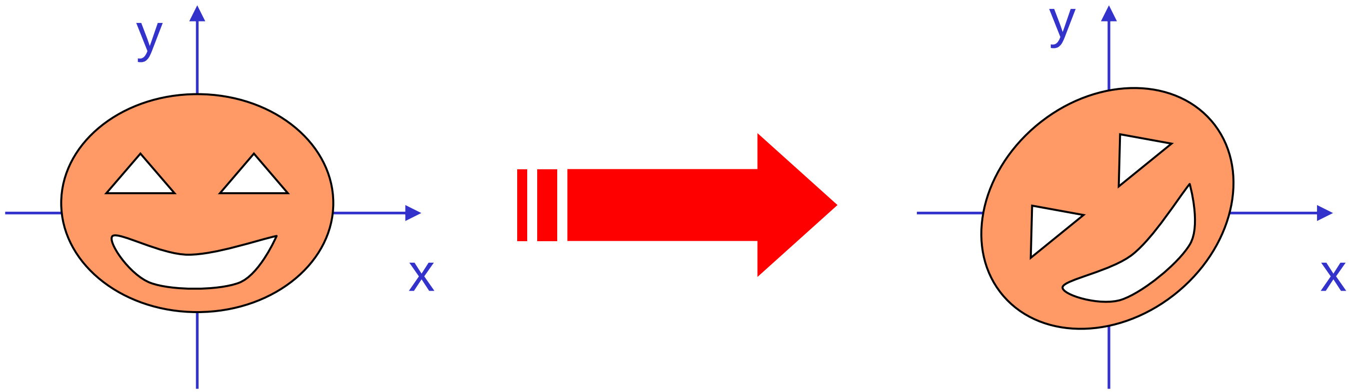
Trasf. di rotazione in 2D
(attorno all'origine, di un angolo β **generico**, in senso antiorario)

Quindi:

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x\cos\beta - y\sin\beta \\ x\sin\beta + y\cos\beta \end{pmatrix}$$

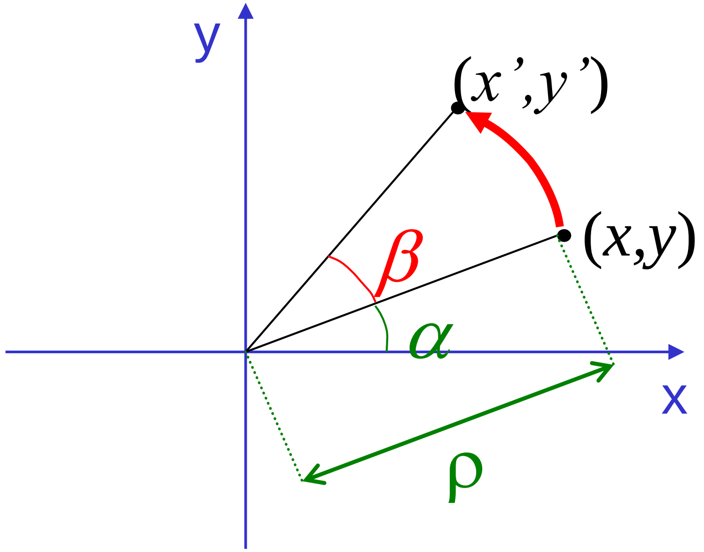
alla fine, il passaggio alle coord polari non serve :-)
bastano il valore di seno e coseno dell'angolo di rotazione β



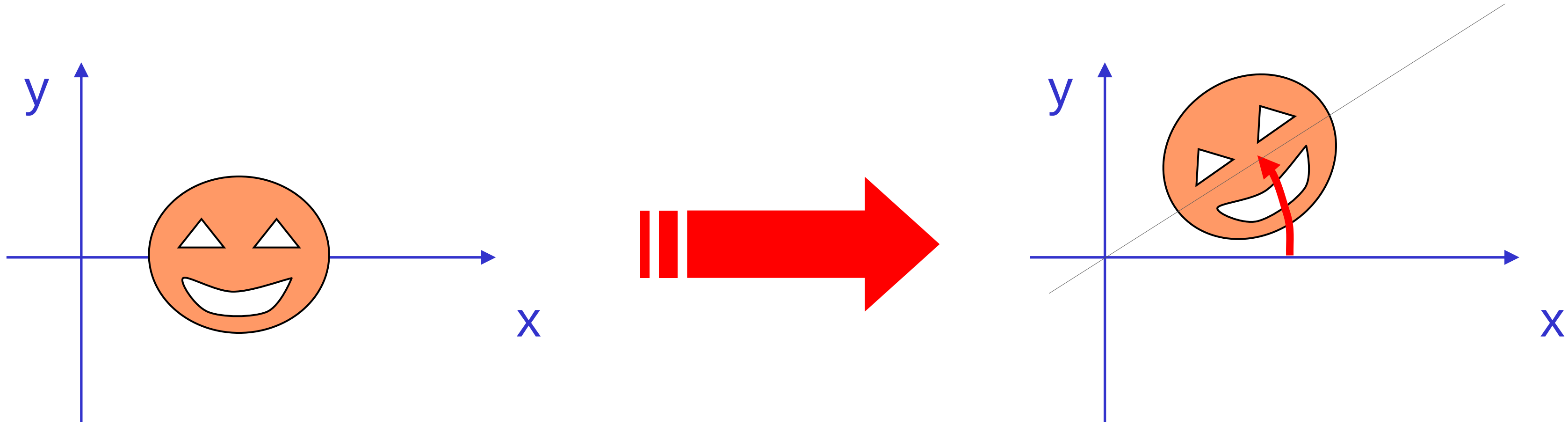


Trasf. di rotazione in 2D (attorno all'origine, di un angolo β **generico**, in senso antiorario)

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x\cos\beta - y\sin\beta \\ x\sin\beta + y\cos\beta \end{pmatrix}$$

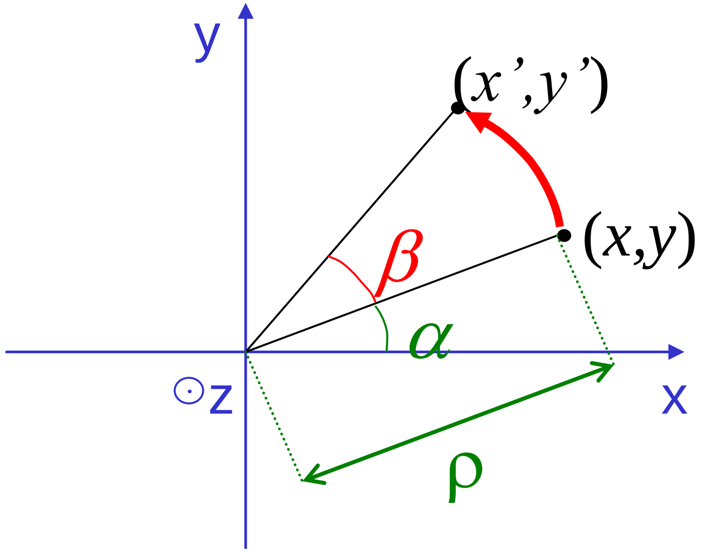


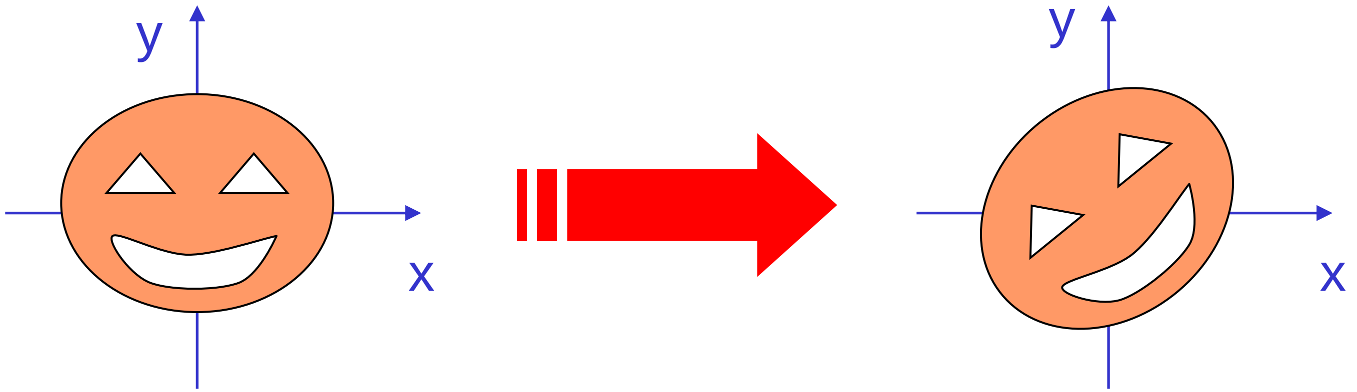
Nota: questa trasformazione ruota attorno all'origine degli assi (non certo attorno al centro delle figure)



Trasf. di rotazione 3D attorno all'asse z (di un angolo β)

$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos\beta - y\sin\beta \\ x\sin\beta + y\cos\beta \\ z \end{pmatrix}$$



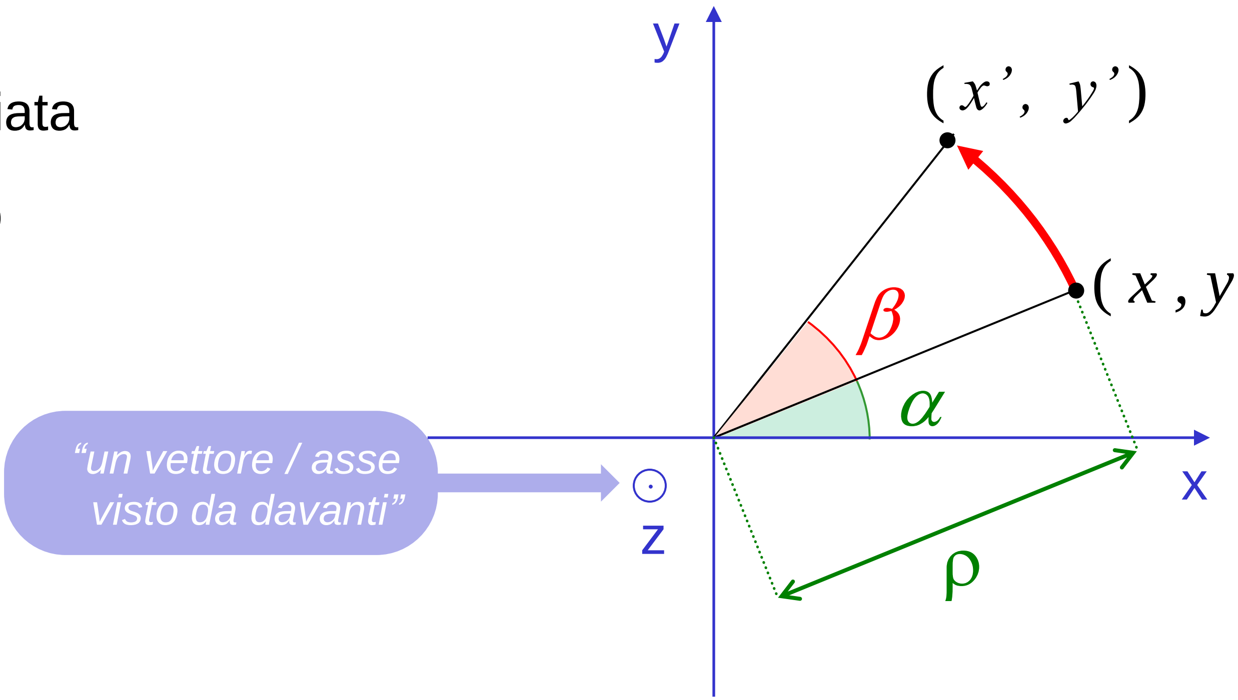


Torniamo in 3D:

Trasf. di rotazione attorno all'asse z (di un angolo β)

z rimane invariata

x e y ruotano



$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos\beta - y\sin\beta \\ x\sin\beta + y\cos\beta \\ z \end{pmatrix}$$

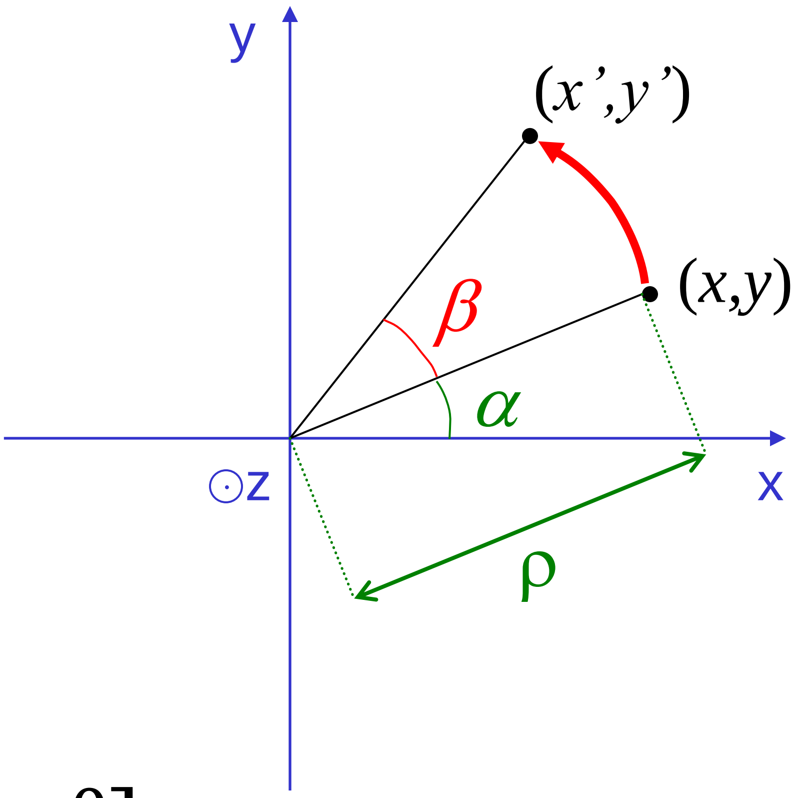
Rotazione 3D attorno all'asse z…
come moltiplicazione con matrice

$$x' = x\cos\beta - y\sin\beta$$

$$y' = x\sin\beta + y\cos\beta$$

$$z' = z$$

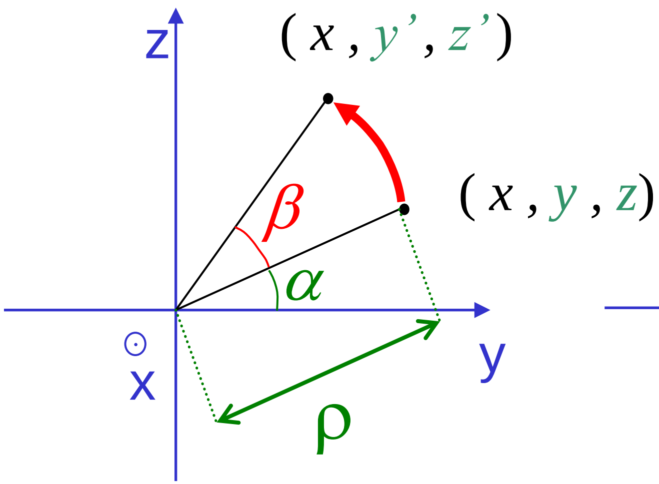
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = R_{Z(\beta)} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\beta - y\sin\beta \\ x\sin\beta + y\cos\beta \\ z \\ 1 \end{bmatrix}$$



$$R_{Z(\beta)} = \begin{bmatrix} \cos\beta & -\sin\beta & 0 & 0 \\ \sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

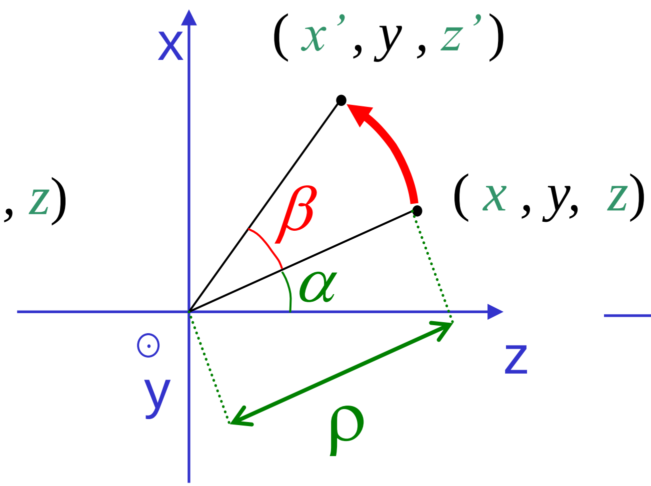
Trasf. di rotazione attorno ad uno dei tre assi

Attorno ad
ASSE X



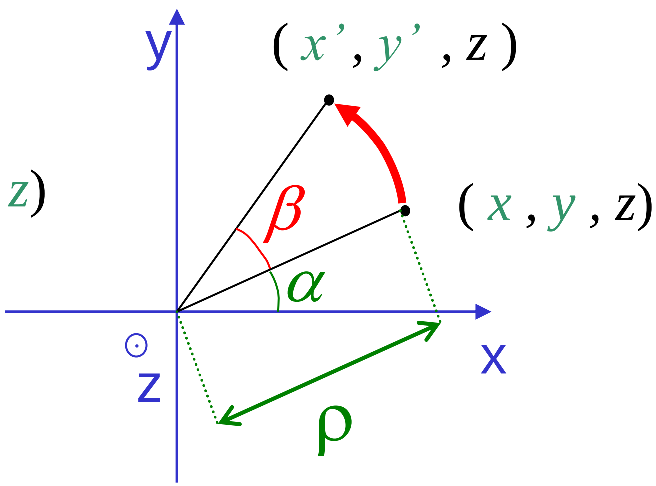
$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \\ y\cos\beta - z\sin\beta \\ y\sin\beta + z\cos\beta \end{pmatrix}$$

Attorno ad
ASSE Y



$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} z\sin\beta + x\cos\beta \\ y \\ z\cos\beta - x\sin\beta \end{pmatrix}$$

Attorno ad
ASSE Z



$$f\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x\cos\beta - y\sin\beta \\ x\sin\beta + y\cos\beta \\ z \end{pmatrix}$$

Matr di Rotazione attorno all'asse x, y, o z

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_Z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

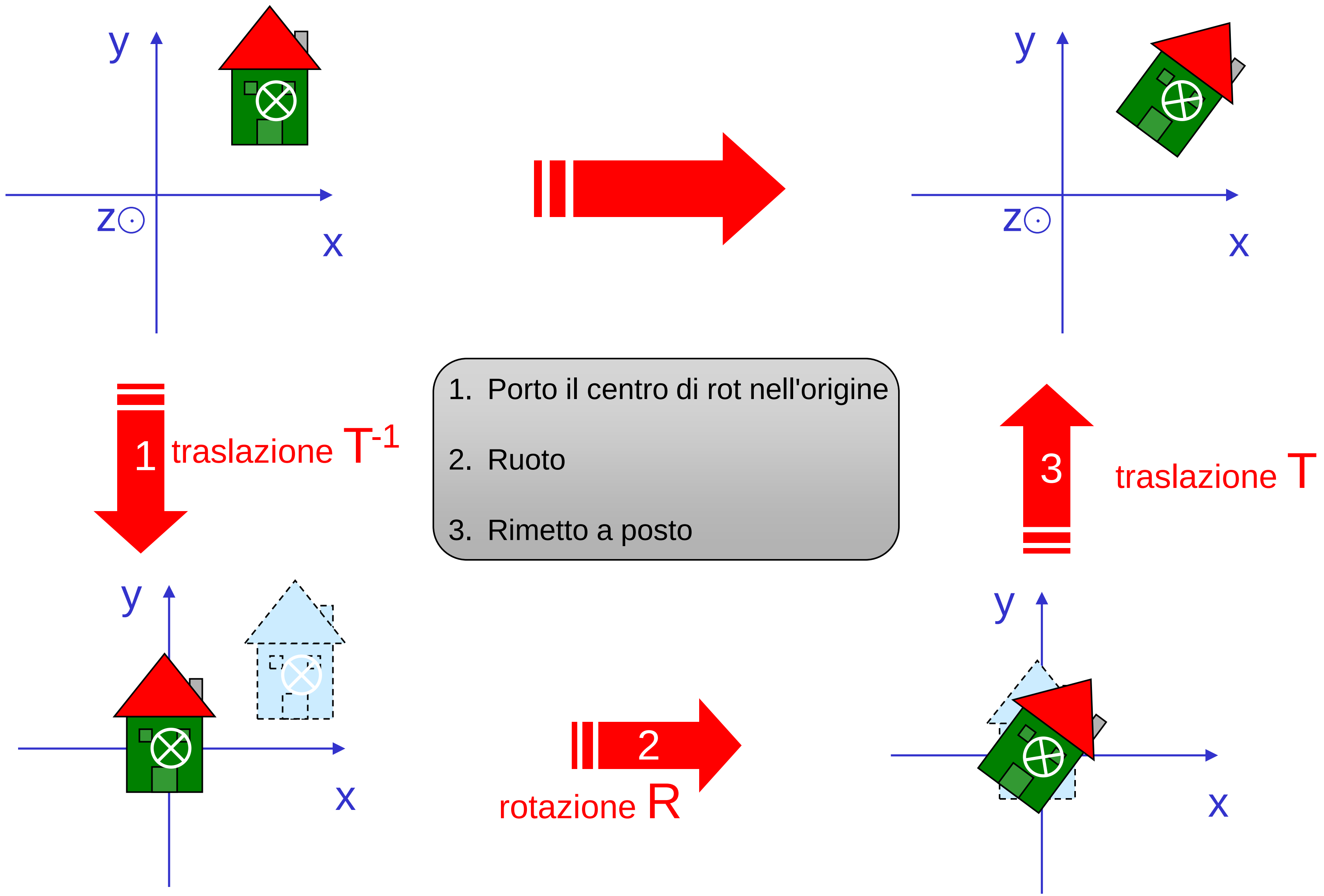
e le inverse?

$$\boxed{R_X(\theta)^{-1} = R_X(-\theta) = R_X(\theta)^{\mathrm{T}}}$$

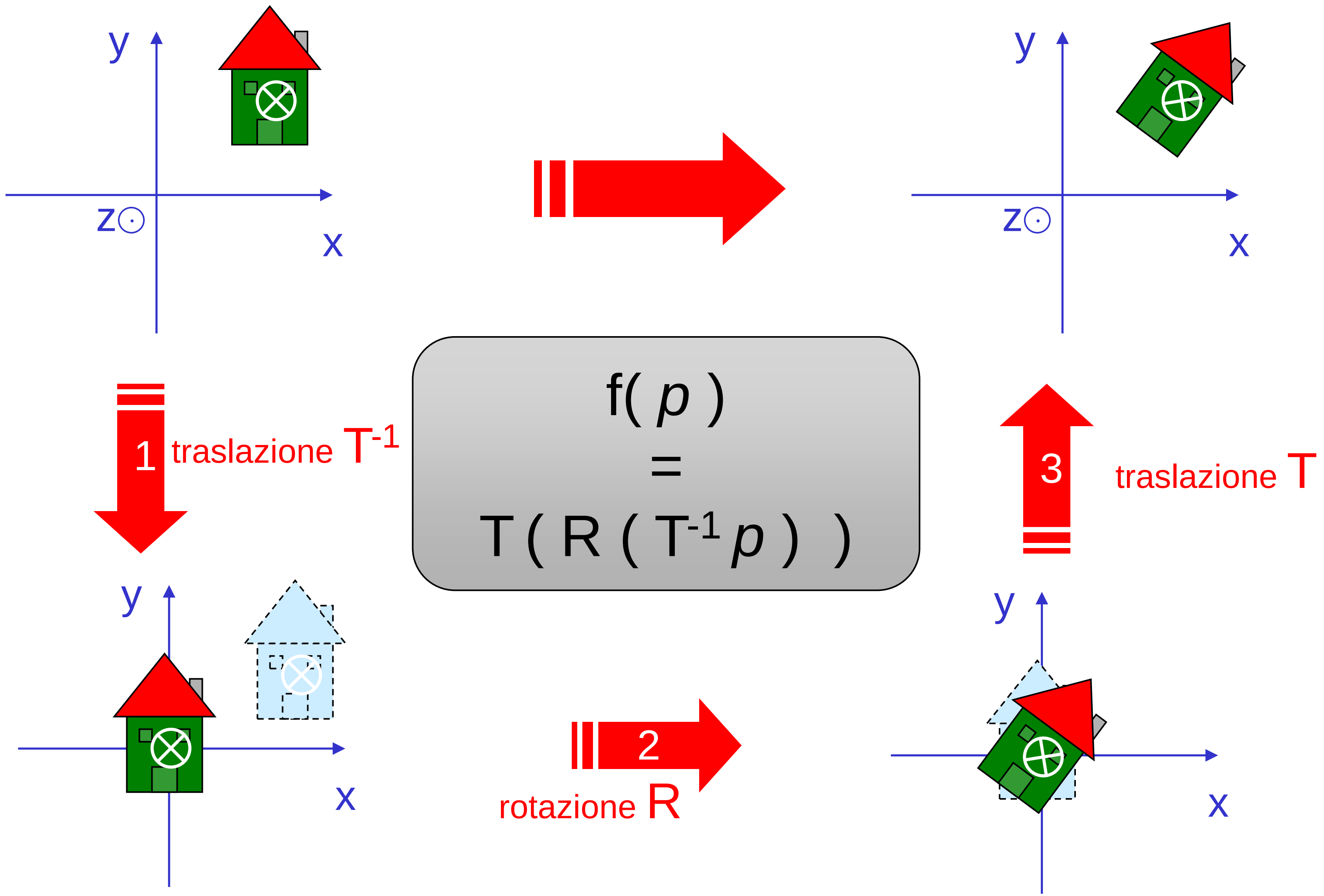
Matrici di rotazione - recap

- ✓ Tutte le matrici di rotazione con asse qualsiasi passante per l’origine le posso costruire componendo rotazioni sui tre assi X, Y, Z
- ✓ Loro inversa: la trasposta
- ✓ Sono matrici ortonormali

Rotazione intorno ad un asse *parallelo* all'asse z



Rotazione intorno ad un asse *parallelo* all'asse z



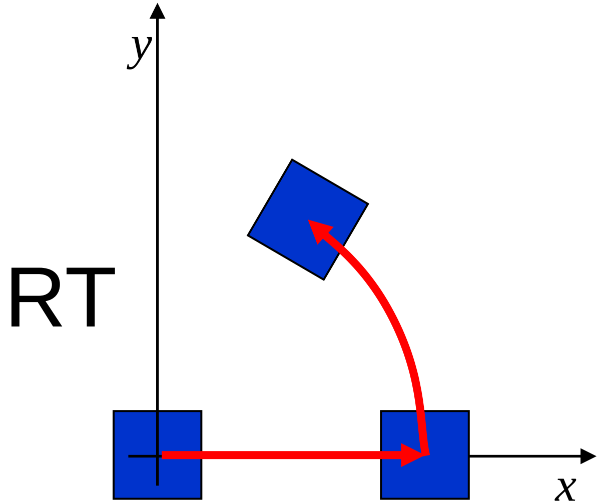
Ripassino: moltiplicazione matrice matrice

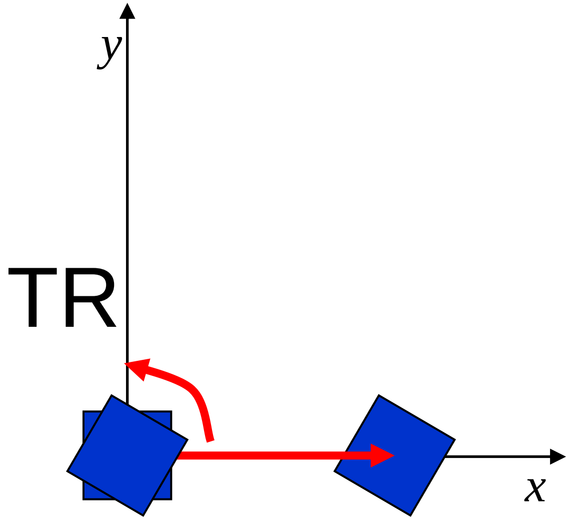
- Attenzione all'inversione: $(AB)^{-1} = B^{-1}A^{-1}$

- Associativa si, ma commutativa no!

  $AB \neq BA$

  ⇒ previsione:
  determinare il corretto ordine delle trasformazioni non sarà intuitivo





Inversione di una affine generica

nb:
$(A\,B)^{-1} = B^{-1}\,A^{-1}$

$$\begin{bmatrix} M & t \\ 0 & 1 \end{bmatrix}^{-1} = \left(\begin{bmatrix} I & t \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} M & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} =$$

$$= \begin{bmatrix} M^{-1} & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} I & -t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} M^{-1} & \\ 0 & 1 \end{bmatrix}$$

$$M^{-1} * -t$$

Sistema di riferimento
o reference *frame* oppure *spazio*

- Definito da
  - ⇒ una base vettoriale { $\mathbf{a}_x$, $\mathbf{a}_y$, $\mathbf{a}_z$ } (assi dello spazio)
  - ⇒ un punto di *origine* $\mathbf{o}$

- Posso esprimere (univocamente) ogni punto **p** come:

$$\mathbf{p} = \mathbf{a}_x x + \mathbf{a}_y y + \mathbf{a}_z z + \mathbf{o}$$

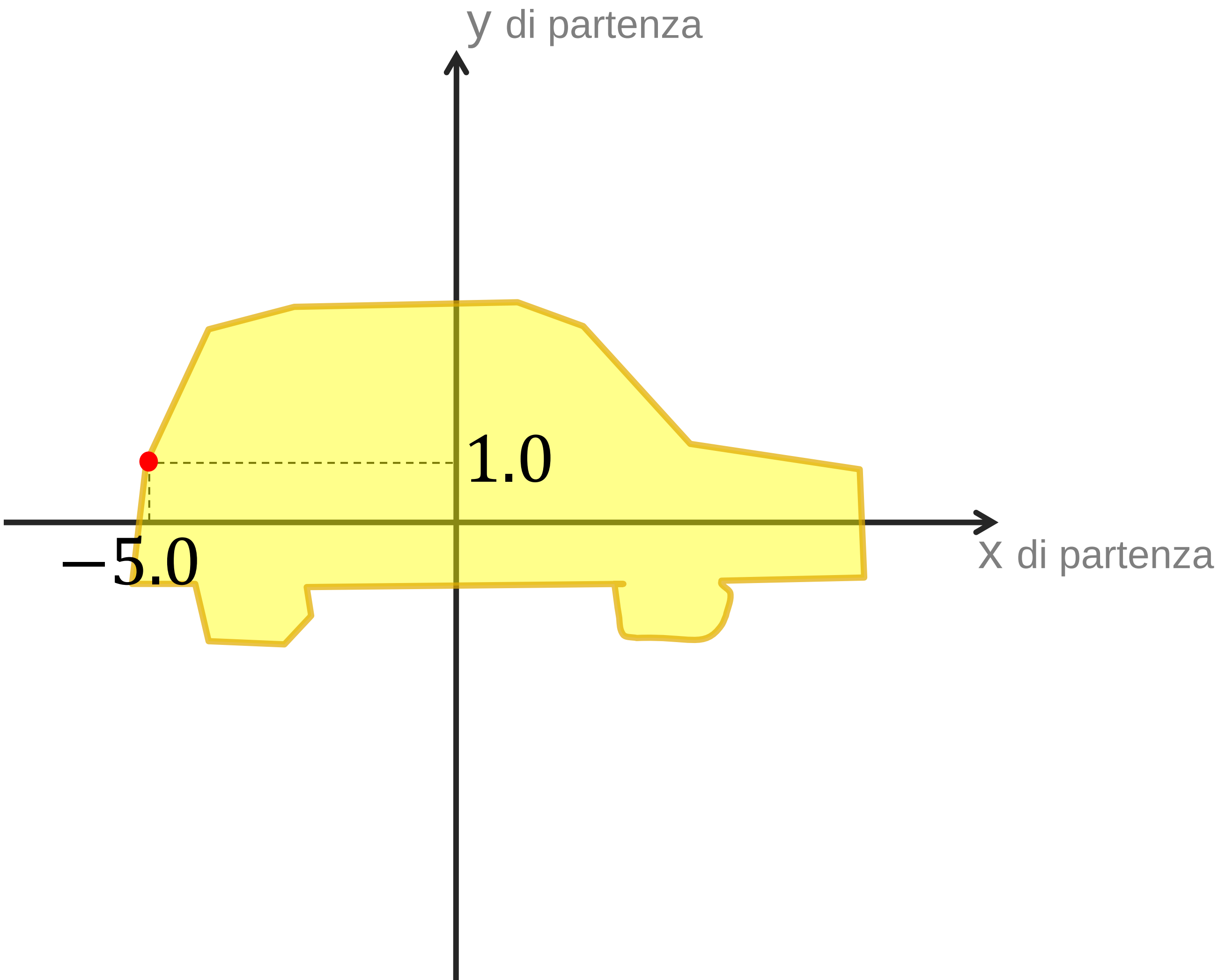
- cioè:

$$\mathbf{v} = \begin{bmatrix} \mathbf{a}_x & \mathbf{a}_y & \mathbf{a}_z & \mathbf{o} \end{bmatrix} \left.\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}\right\} \boxed{\text{coordinate } \textit{omogenee} \text{ di } \mathbf{p}}$$

Es: rot di 45° su Z

$$\begin{bmatrix} +0.7 & -0.7 & 0 & 0 \\ +0.7 & +0.7 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \end{bmatrix} \begin{bmatrix} -5.0 \\ 1.0 \\ 0.0 \\ 1 \end{bmatrix} = \begin{bmatrix} -4.4 \\ -2.8 \\ 0.0 \\ 1 \end{bmatrix}$$

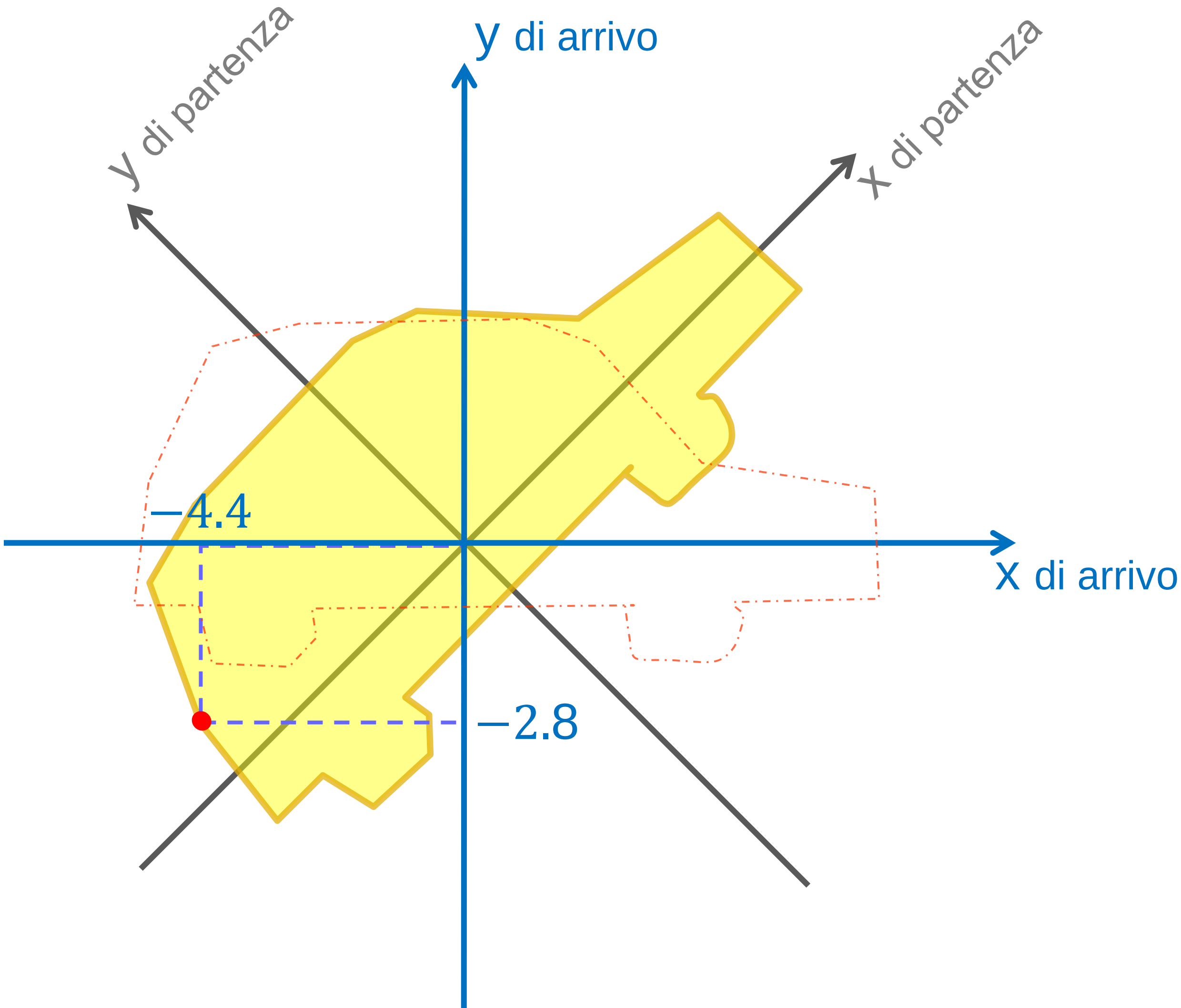
matrice per passare dal frame di partenza al frame di arrivo



Es: rot di 45° su Z

$$\begin{bmatrix} +0.7 & -0.7 & 0 & 0 \\ +0.7 & +0.7 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \end{bmatrix} \begin{bmatrix} -5.0 \\ 1.0 \\ 0.0 \\ 1 \end{bmatrix} = \begin{bmatrix} -4.4 \\ -2.8 \\ 0.0 \\ 1 \end{bmatrix}$$

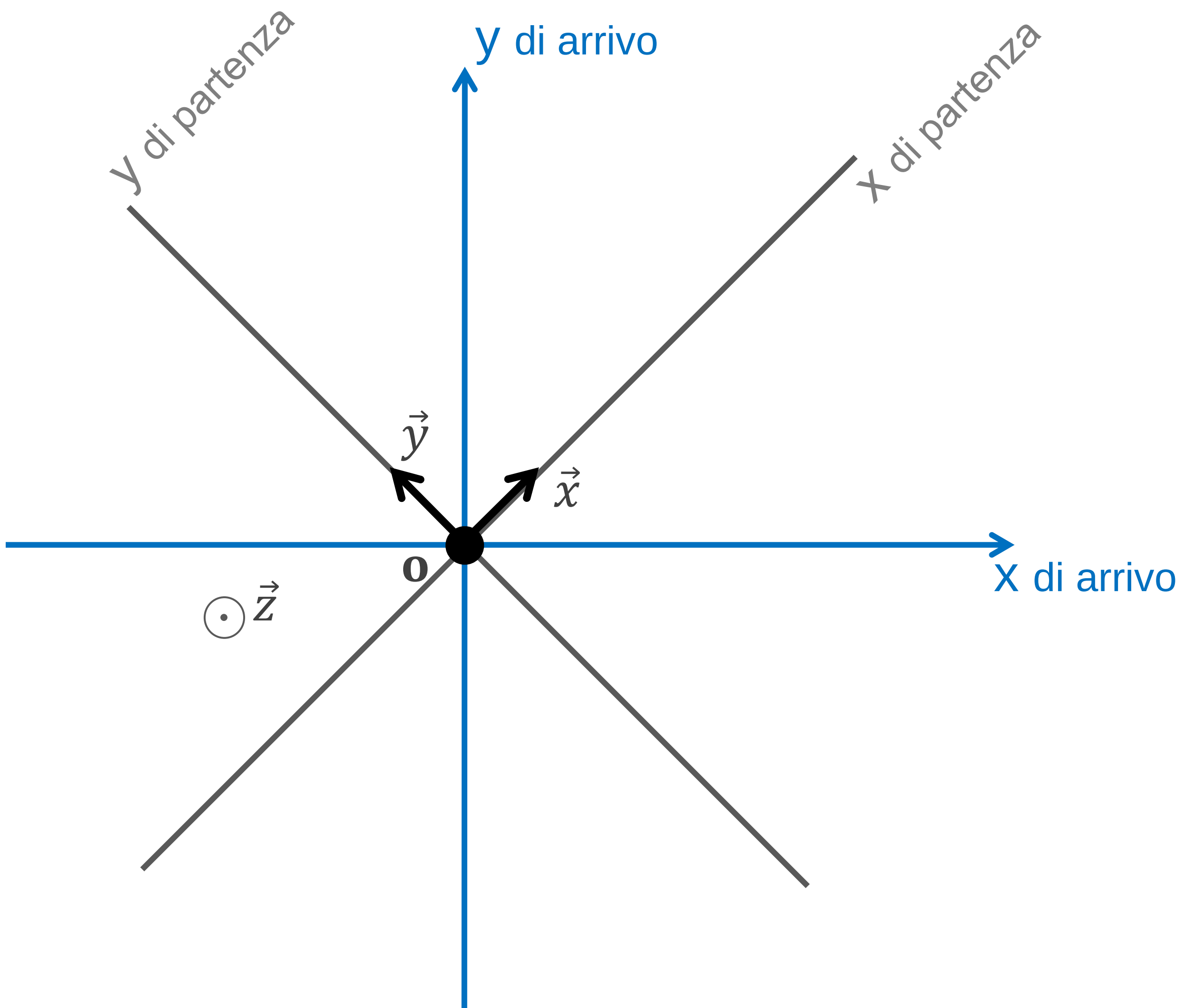
matrice per passare dal frame di partenza al frame di arrivo



Es: rot di 45° su Z

$$\begin{array}{c}\begin{bmatrix} +0.7 & -0.7 & 0 & 0 \\ +0.7 & +0.7 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \end{bmatrix} \\ \begin{array}{cccc} \vec{x} & \vec{y} & \vec{z} & \mathbf{o} \end{array}\end{array} \begin{bmatrix} -5.0 \\ 1.0 \\ 0.0 \\ 1 \end{bmatrix} = \begin{bmatrix} -4.4 \\ -2.8 \\ 0.0 \\ 1 \end{bmatrix}$$

matrice per passare dal frame
di partenza al frame di arrivo



Rotations

Rotations as 3x3 matrices (9 scalars)

Eigendecomposition of Rotation matrix?

-One real eigenvalue $\lambda=1$
-What is the meaning of its associate eigenvector?

$$\mathbf{Rv=\lambda v=v}$$

Euler angles

Rotations as Euler angles (3 scalars)

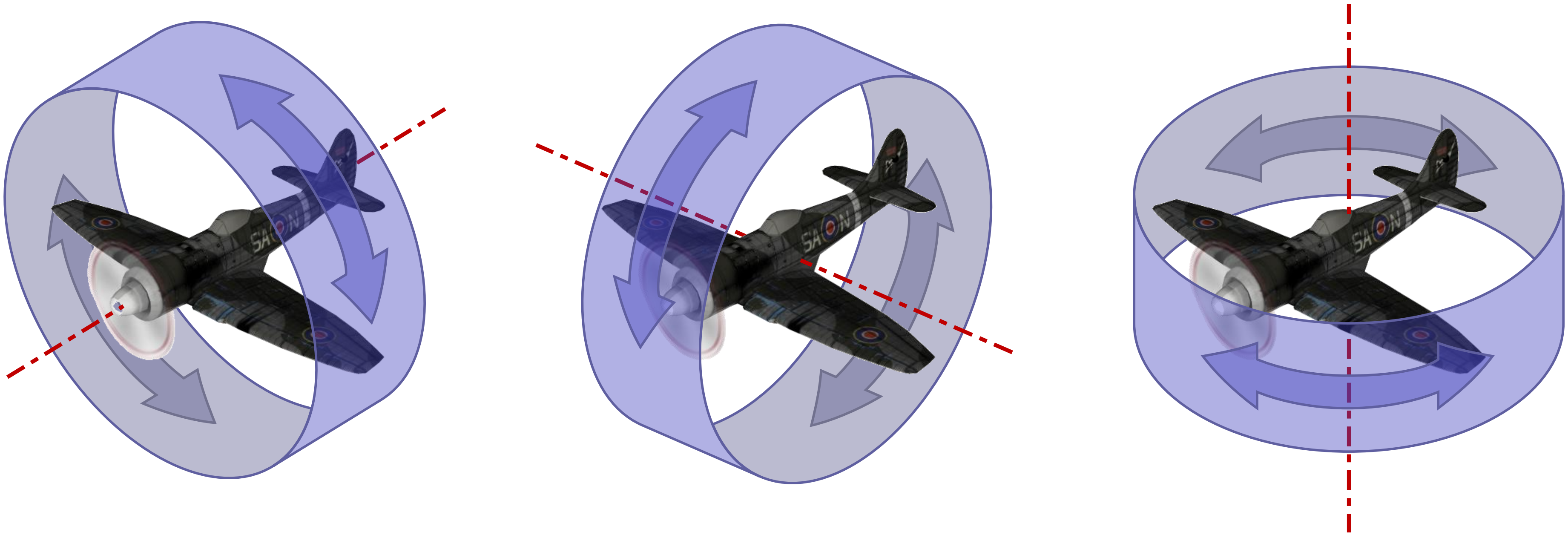
- ✓ Any 3D rotation can be expressed as:
  - ⇒a rotation around X axis (by α degrees), followed by:
  - ⇒a rotation around Y axis (by β degrees), followed by :
  - ⇒a rotation around Z axis (by γ degrees):

this order (X-Y-Z) is chosen arbitrary but once and for all! (in a given game engine / lib)

- ✓ Angles α β γ : "Euler angles" of a specific rotation
  - ⇒(therefore: its "coordinates")

Rotations as Euler angles (3 scalars)

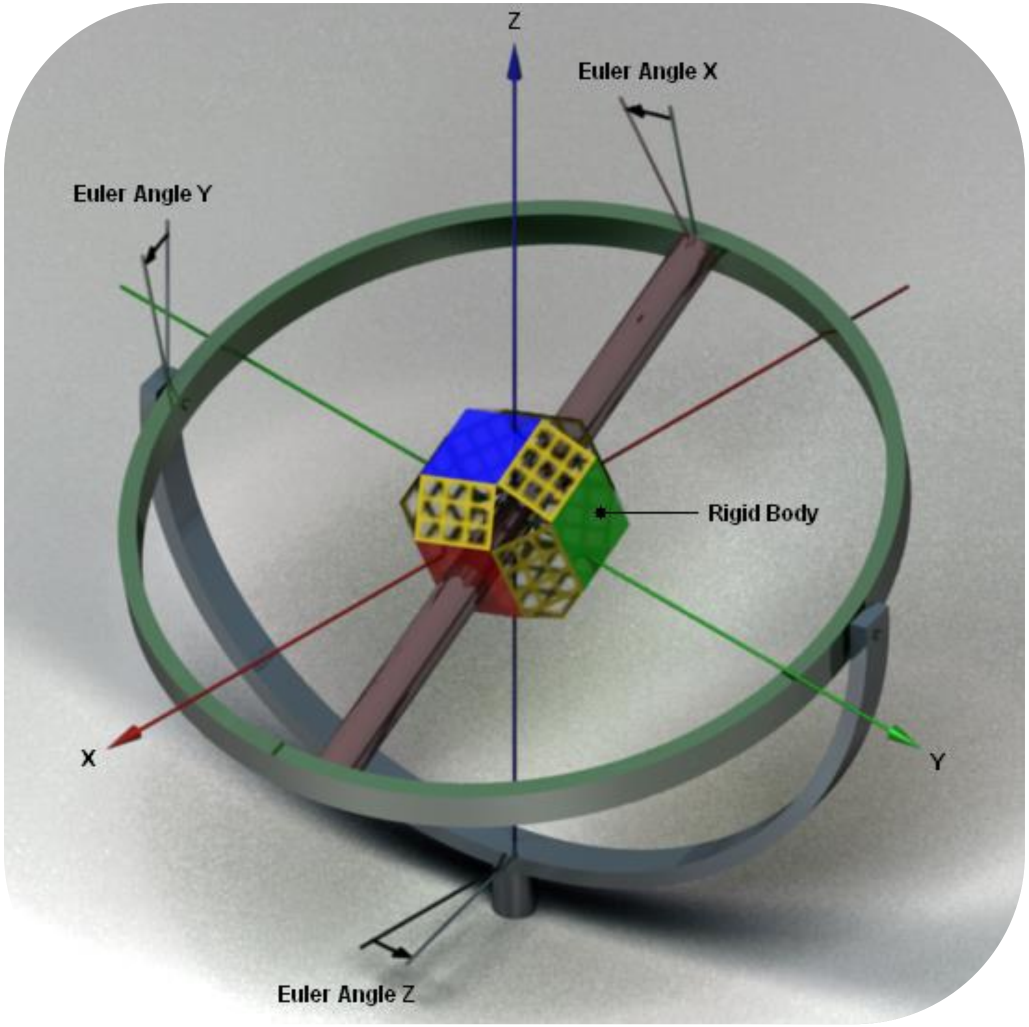
✓ In nautical / aeronautical language, the three angles have names:



roll (*rollio*) — pitch (*beccheggio*) — yaw (*imbardata*)

Rotations as Euler angles (3 scalars)

- ✓ A physical implementation: “three axes globe”

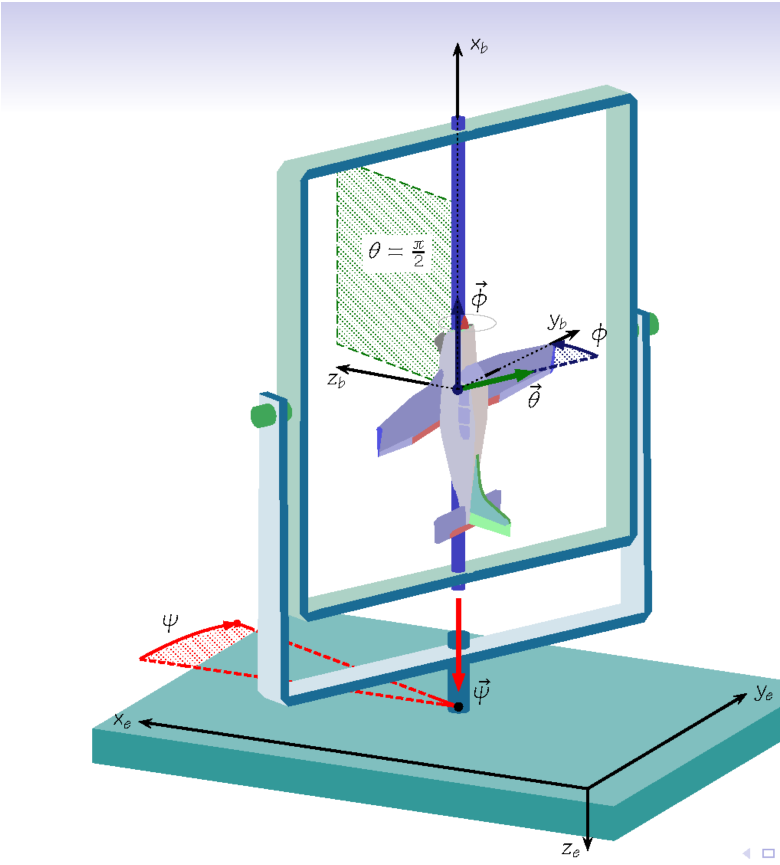


Rotations as Euler angles (3 scalars)

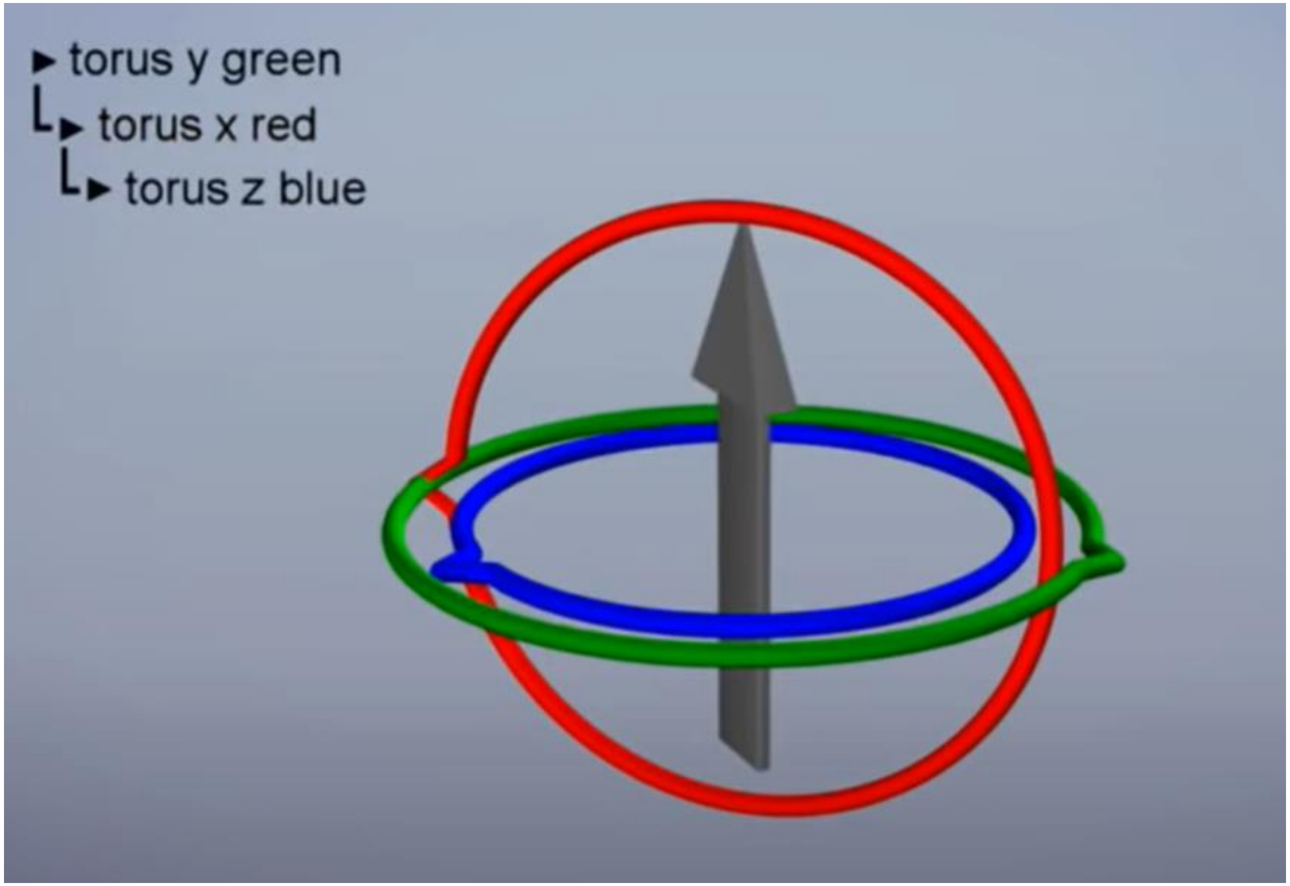
- ✓ Is it 1:1 ?
  - ⇒ 1 rotation ⇔ 1 euler angle triplet ?
- ✓ Almost
  - ⇒ assuming angles are properly bounded

Ugly exception:
"**GIMBAL LOCK**"

- ⇒ when 1st rotation makes the axes of the next two axes *coincide*
- ⇒ this cannot be avoided, no matter how axes are chosen



Gimbal lock



- ✓ https://www.youtube.com/watch?v=zc8b2Jo7mno

Rotations as Euler angles (3 scalars)

- ✓ Conciseness: perfect! 3 scalars for 3 DOF
- ✓ Application : a bit work-intensive
  - ⇒ three rotations in succession
- ✓ Interpolation : you can do that…
  - ⇒ just interpolate the three angles
  - ⚠ ⇒ (remember to always "pick the *shortest path*" whenever interpolating angles: that is, must take in account the $\alpha \approx \alpha + 360\,k$ equivalence)

  …but results won't always be nice !
- ✓ Composite / invert: not easy nor immediate…

from: euler angles
to: 3x3 matrix

✓ Easy to write down!

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$\mathrm{M} = \mathrm{R_z}(\gamma) \cdot \mathrm{R_y}(\beta) \cdot \mathrm{R_x}(\alpha)$$

⇒ requires several sin / cos evaluations (and matrix mult)

✓ What about the vice-versa?

⇒ not very convenient:
many inverse trigonometric functions

Axis angle

Rotations as axis & angle

- Any rotation can be expressed as:
  - one rotation by some **angle** around some **axis**

must be appropriately chosen

- **Angle:** a scalar
- **Axis:** a versor (3 scalars)
  - note: the axis is considered to pass around the origin. For the more general case, combine with translations.

Rotations as axis & angle

- Compactness: good, 4 scalars
  - Just one more than bare minimum
- Ease of application: not too good ☹
  - Ways include: switch to 3x3 matrix (exercise: how to) or to quaternion: see later
- Invert: super easy / quick
  - just flip the angle sign *or* the axis vector
  - question: what if both?
    answer: Rotation is inverted twice:
    it's back to the same rotation again! 🤔

Rotations as axis & angle: equivalent representations

- Therefore: $(\ a_x\ ,\ a_y\ ,\ a_z\ ,\ \alpha)$
  and $(-a_x, -a_y, -a_z\ , -\alpha)$
  represent the same rotation
- Any rotation has two **equivalent representations** in this format
  - except the identity, which has infinitely many: angle α = 0, with any axis a_x, a_y, a_z
- This is always a bit inconvenient
  - Complicates interpolation ("shortest path" problems)
  - Complicates testing for equality/similarity, etc.

Rotations as axis & angle

- Compositing rotations:
  not at all immediate or easy to do ☹
- Interpolating rotations: very good!
  - Just interpolate axis and angle separately
  - Some *caveat*:
    - 1) *shortest path* for axes: first, flip either rotation (both its axis & angle) when this makes the two axes closer (how to test?)
    - 2) *shortest path* for angles: as usual, angles must then be interpolated… «modulo 360°»,
    - 3) interpolate between axes requires SLERP or NLERP (when interpolating versors)
    - 4) beware degenerate cases (opposite axes); point 1 avoids this
  - best results! Usually produces the "right" rotation

Rotations as axis and angle, variant: as axis angle

- axis: v (versor, $|v| = 1$)
- angle: α (scalar)
- can be represented as one vector v' (3 scalars)

  $v' = \alpha\, v$

  - angle $\alpha = |v'|$
  - axis $v = v' / \alpha$
  - note: when $\alpha = 0$, the axis is lost… it's ok, we don't need it!
- more compact, but fairly equivalent
  - actually, better: we now have only 1 representation per rotation (why?) … including the identity (why?)

Quaternions

A flashback: Complex Numbers in a nutshell 1/3

- It all starts with a «fantasy» assumption, which is: there is an imaginary number i such that $i^2 = -1$
  - And for any other purpose, i behaves just like a (non-zero) Real number
- Consequences:

  real part *imaginary part*

  - We now have number of the form $a + b\ i$, with $a, b \in \mathbb{R}$, called complex numbers (the set is $\mathbb{C}$)
  - The algebra of complex numbers (how to sum, multiply, invert them…) is simply determined by the «fantasy» assumption above

A flashback:
Complex Numbers in a nutshell 2/3

- For example, sum:

real part *imaginary part*

$$(a + b\ i) + (c + d\ i) = (a + c) + (b + d)i$$

- For example, product (remembering $i^2 = -1$):

$$(a + b\ i) * (c + d\ i) = (ac - bd) + (ad + bc)i$$

- For example, inverse (check):

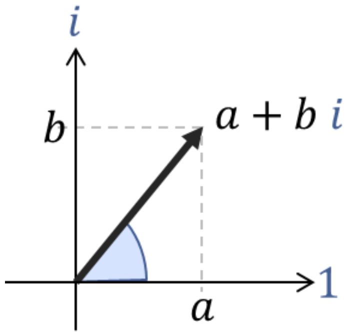
$$(a + b\ i)^{-1} = \frac{(a\ - b\ i)}{a^2 + b^2}$$

the «coniugate» of (a + b i)

the squared «magnitude» of (a + b i)

- What is interesting to us is the **geometric interpretation** of these objects & operations

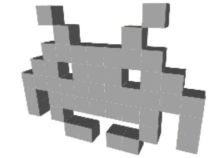
A flashback: Complex Numbers in a nutshell 3/3



- Geometric interpretation:
  - $a + b\,i$ represents the vector/point (a, b)
  - Complex sum is vector sum
  - Complex conjugate is mirroring with the Real axis (horizontal)
  - Product is… add angles (with Real axis), multiply magnitudes
- Therefore,
  - product with a unitary (magnitude = 1) complex number is a pure 2D rotation
  - A complex number $c \in \mathbb{C}$ with $\|c\| = 1$ represents a 2D rot; multiply vector $(x + y\,i)$ with c means to rotate it

Wouldn't it be cool to have the same for 3D rotations?

Quaternions

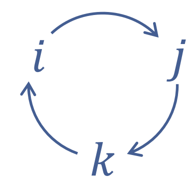


as a table:

×	i	j	k
i	-1	$+k$	$-j$
j	$-k$	-1	$+i$
k	$+j$	$-i$	-1

- New «fantasy» assumption: there are three different "imaginary" numbers i , j , k such that:

$$\begin{cases} i^2 = k^2 = j^2 = -1 \\ ij = k \ , \qquad ji = -k \\ jk = i \ , \qquad kj = -i \\ ki = j \ , \qquad ik = -j \end{cases}$$



  - for any other purpose, i, j, k behave like real numbers
- Consequences:
  - We now have number of the form $\underbrace{a\, i + b\, j + c\, k}_{\text{imaginary parts}} + \underbrace{d}_{\text{real part}}$, with $a, b, c, d \in \mathbb{R}$, called Quaternions (their set is $\mathbb{H}$)
  - The algebra of quaternions (how to sum, multiply, invert them…) is simply determined by the «fantasy» assumption
  - Again, what is interesting to us is the **geometric interpretation**…

Quaternions: how to write them (equivalently)

- Algebraic form: $a\, i + b\, j + c\, k + d$
  - often, omitting the zeros, e.g. $i + 2\, k$ is a quaternion
- As vectors of $\mathbb{R}^4$: $(a\,,\ b\,,\ c\,,\ d\,)$
- As vector & scalar pair: $(\vec{v}\ ,\ d)$

imaginary part, a vector $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ *real part, a scalar*

- Conjugate of a quaternion: invert the sign of the imaginary part

Quaternions: operations how-to

$$q \in \mathbb{H} \qquad\qquad q = ai + bj + ck + d$$

- Sum, Scale, Interpolate , etc.: trivial
  - same as 4D vectors
- Magnitude

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2}$$

$$\|q\|^2 = a^2 + b^2 + c^2 + d^2$$

  - «unitary» if it's 1
  - same as 4D vectors

Quaternions: operations how-to

$$\mathrm{q} \in \mathbb{H} \qquad\qquad \mathrm{q} = ai + bj + ck + d$$

- Product: just apply «fantasy» assumptions
  - Observe: product is not commutative (nor anticommut.)
  - (see next 3 slides for the math)

- «Coniugate»:

  Flip imaginary parts

  - like for complex numbers: $\bar{\mathrm{q}} = -ai - bj - ck + d$

- Inverse: (like for complex numbers) $\mathrm{q}^{-1} = \bar{\mathrm{q}} \,/\, \|\mathrm{q}\|^2$
  - For unitary quat, it's just the coniugate

Quaternion Product

×	a i	+	b j	+	c k	+	d	
e i		+		+		+		+
+								
f j		+		+		+		+
+								
g k		+		+		+		+
+								
h		+		+		+		

Quaternion Product

$\vec{v}$

×	a i	+	b j	+	c k	+	d	
e i	-1 ae	+	k be	+	$-j$ ce	+	i de	+
+								
f j	$-k$ af	+	-1 bf	+	i cf	+	j df	+
+								
g k	j ag	+	$-i$ bg	+	-1 cg	+	k dg	+
+								
h	i ah	+	j bh	+	k ch	+	hd	

$\vec{w}$

$(\vec{w}, h)$

$\cdot$

$(\vec{v}, d)$

$=$

(some vector

,

some scalar)

Quaternion Product

$\vec{v}$

×	$a\ i$	+	$b\ j$	+	$c\ k$	+	d	
$\vec{w}$: $e\ i$	-1 ae	+	k be	+	$-j$ ce	+	i de	+
+								
$f\ j$	$-k$ af	+	-1 bf	+	i cf	+	j df	+
+								
$g\ k$	j ag	+	$-i$ bg	+	-1 cg	+	k dg	+
+								
h	i ah	+	j bh	+	k ch	+	hd	

$$(\vec{w}, h) \cdot (\vec{v}, d) = (\vec{w}\, d + \vec{v}\, h + \vec{w} \times \vec{v},\; h\, d - \vec{w} \cdot \vec{v})$$

Quaternions: Geometric Interpretation!

- A quaternion $\mathbf{q} = (\vec{\mathrm{v}}, d)$ represents :
  - the 3D point or vector $\vec{\mathrm{v}}$, when $d = 0$
  - a 3D rotation, when $\mathbf{q}$ is unit, i.e. $||\mathrm{q}||^2 = ||\vec{\mathrm{v}}||^2 + d^2 = 1$
  - (neither, otherwise)
- If $\mathbf{q}$ is a rotation and $\mathbf{p}$ is a point ($\mathbf{q}, \mathbf{p} \in \mathbb{H}$) then...
  - $\mathbf{q} \cdot \mathbf{p} \cdot \bar{\mathbf{q}}$ is the rotated point / vector
  - $\bar{\mathbf{q}}$ is the inverse rotation
  - $\mathbf{q}_0 \cdot \mathbf{q}_1$ is the composited rotation (first $\mathbf{q}_1$ then $\mathbf{q}_0$)
  - (so, $\bar{\mathbf{q}} \cdot \mathbf{p} \cdot \mathbf{q}$ is the pt rotated... in the *other* direction)

Compositing Quaternions: why it works

$q_0 , q_1 , p \ \in \mathbb{H}$

q_0 , q_1 represent rotations

p represents a point

$$\overbrace{q_0 \cdot \overbrace{(q_1 \cdot p \cdot \bar{q}_1)}^{\text{p rotated by q1}} \cdot \bar{q}_0}^{\text{p rotated by q1, rotated by q0}}$$

product is associative
(like for complex numbers)

$$=$$

$$(q_0 \cdot q_1) \cdot p \cdot (\bar{q}_1 \cdot \bar{q}_0)$$

$\bar{r} \cdot \bar{s} = \overline{s \cdot r}$
(rules of quaternions)
(remember: product is not commutative)

$$=$$

$$(q_0 \cdot q_1) \cdot p \cdot \overline{(q_0 \cdot q_1)}$$

3D Rotations as Quaternions

- quaternion $\mathbf{q}$ representing the 3D rotation of angle α around axis $\hat{\mathrm{a}}$:
  - $\mathrm{q} = \left(\sin\left(\frac{\alpha}{2}\right) \hat{\mathrm{a}} \, , \; \cos\left(\frac{\alpha}{2}\right) \right)$

  that is
  - $\mathrm{q} = \sin\left(\frac{\alpha}{2}\right) \hat{\mathrm{a}}_x i + \sin\left(\frac{\alpha}{2}\right) \hat{\mathrm{a}}_y j + \sin\left(\frac{\alpha}{2}\right) \hat{\mathrm{a}}_z k + \cos\left(\frac{\alpha}{2}\right)$
- Observe that $||\mathrm{q}||^2 = 1$

verify

3D Rotations as Quaternions: a problem

- Around axis $\hat{a}$ by angle α :

$$q = \left(\sin\left(\frac{\alpha}{2}\right)\hat{a} \, , \cos\left(\frac{\alpha}{2}\right) \right)$$

- Around axis $-\hat{a}$ by angle $(-\alpha)$: (it's the same rotation!)

$$q' = \left(-\sin\left(\frac{-\alpha}{2}\right)\hat{a} \, , \cos\left(\frac{-\alpha}{2}\right) \right) = q$$

same quaternion :-)

Good! But:

- Around axis $\hat{a}$ by angle $(\alpha + 360°)$: (it's the same rotation!)

$$q'' = \left(\sin\left(\frac{\alpha}{2} + 180°\right)\hat{a} \, , \cos\left(\frac{\alpha}{2} + 180°\right) \right) =$$

$$= \left(-\sin\left(\frac{\alpha}{2}\right)\hat{a} \, , -\cos\left(\frac{\alpha}{2}\right) \right) = -q$$

different quaternion :-(

- Conclusion:
  quaternion q and quaternion $-q$ encode the same rotation

3D Rotations as Quaternions: a problem

Given a quaternion which is a rotation:

- Flip its real part: invert rotation
- Flip its imaginary part (conjugate): same
- Flip everything: same rotation

Every rotation is encoded
by two different quaternions $\mathbf{q}$ and $-\mathbf{q}$.

Interpolating two quaternions representing rotations

Good results, but two *caveats*:

⚠ Take the "shortest path" (as usual):
flip 2nd quaternion first, if this makes them closer

- Distance defined as dot product in 4D
  (they are 4D unit vectors!)

⚠ Loss of normality

- Needs re-normalization (NLERP),
- Or SLERP
  (again, consider them as 4D unit vectors)

Quaternions as rotations

- Almost as compact as possible to store (4 scalars)
- Trivial to invert
- Fast to composite
- Fast to apply
- Easy to ensure they are still rotations (just normalize)
  - Even after long sequences of cumulations, unlike matrices
- Behaves well under interpolation
  - Even with just NLERP – better with SLERP

Exponential matrix