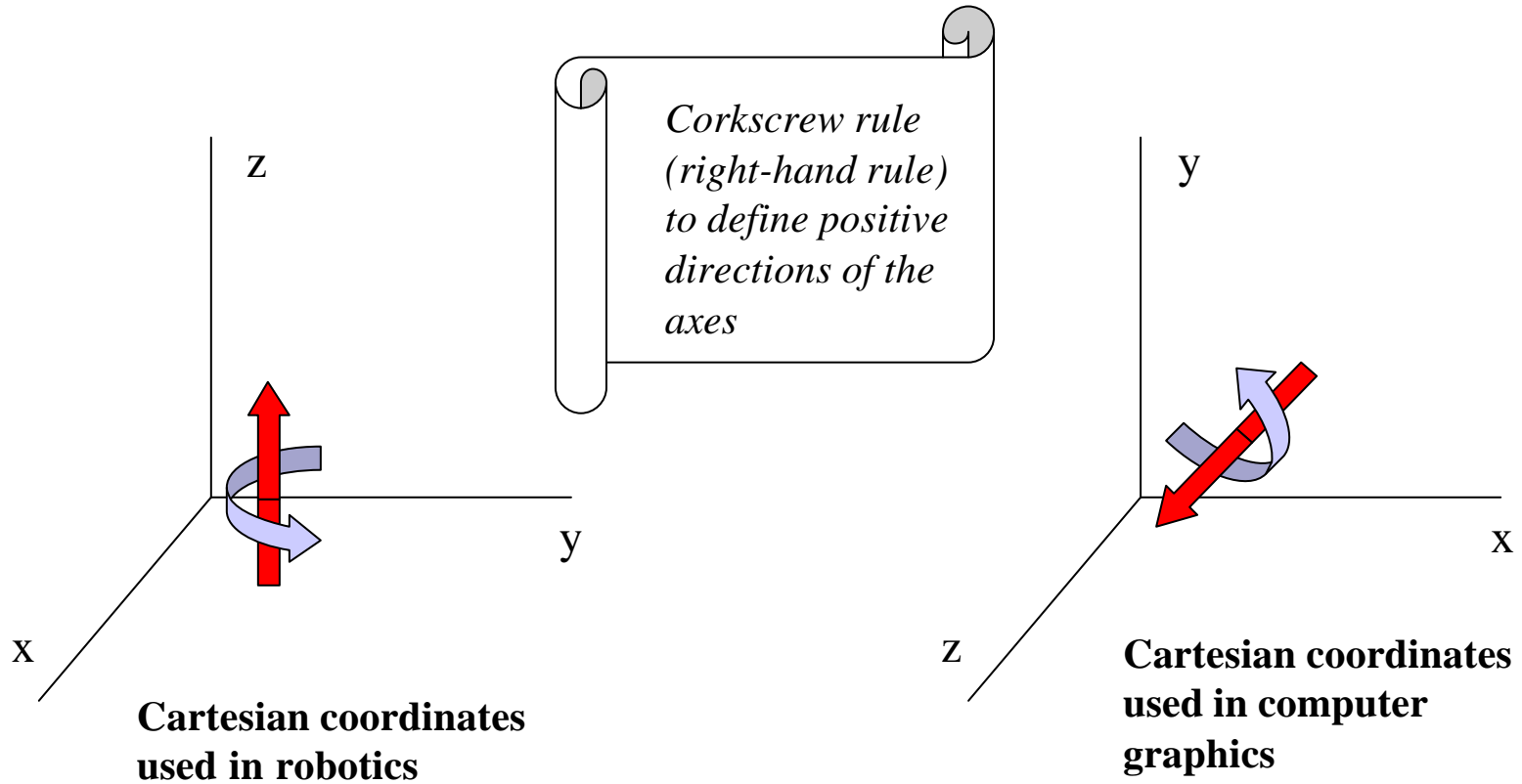

OBJECT LOCATION

* CARTESIAN COORDINATES



* RECOVERING THE LOCATION OF MOVING OBJECTS: *TRANSLATION*

Using a reference frame N affixed to the object allows the calculation of the variable location of the point Q on the moving object in a given reference frame R as the sum of two vectors, a constant one representing the location of Q relative to N and another variable representing the location of N relative to R:

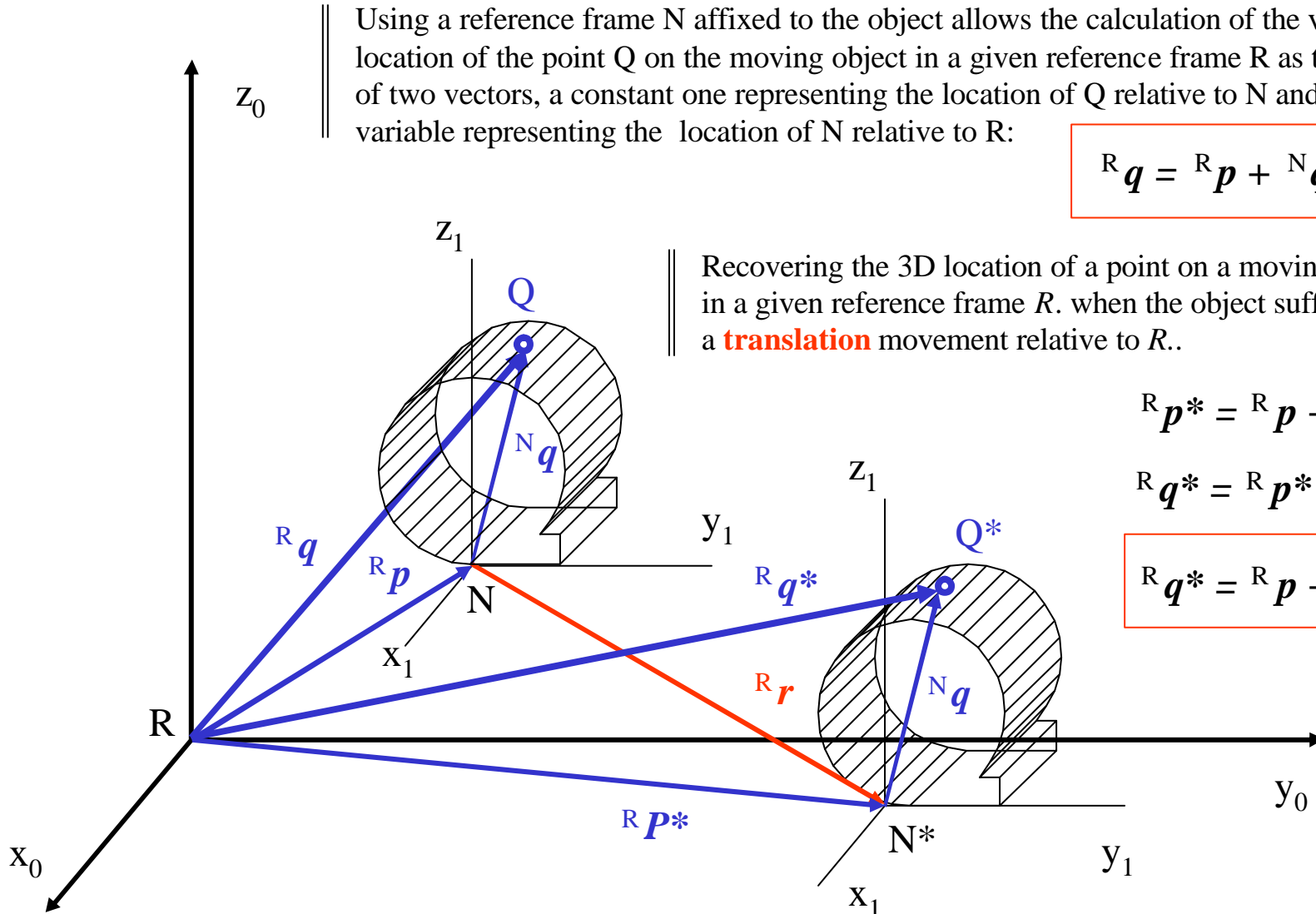
$${}^R q = {}^R p + {}^N q$$

Recovering the 3D location of a point on a moving object in a given reference frame R. when the object suffers only a **translation** movement relative to R..

$${}^R p^* = {}^R p + {}^R r$$

$${}^R q^* = {}^R p^* + {}^N q$$

$${}^R q^* = {}^R p + {}^N q + {}^R r$$

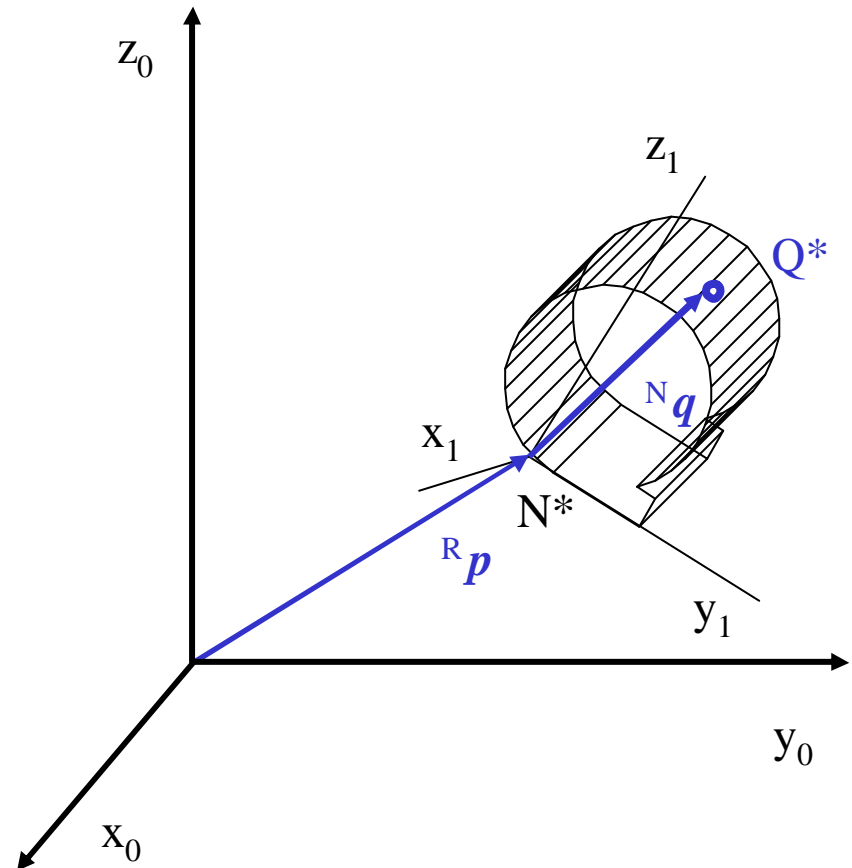
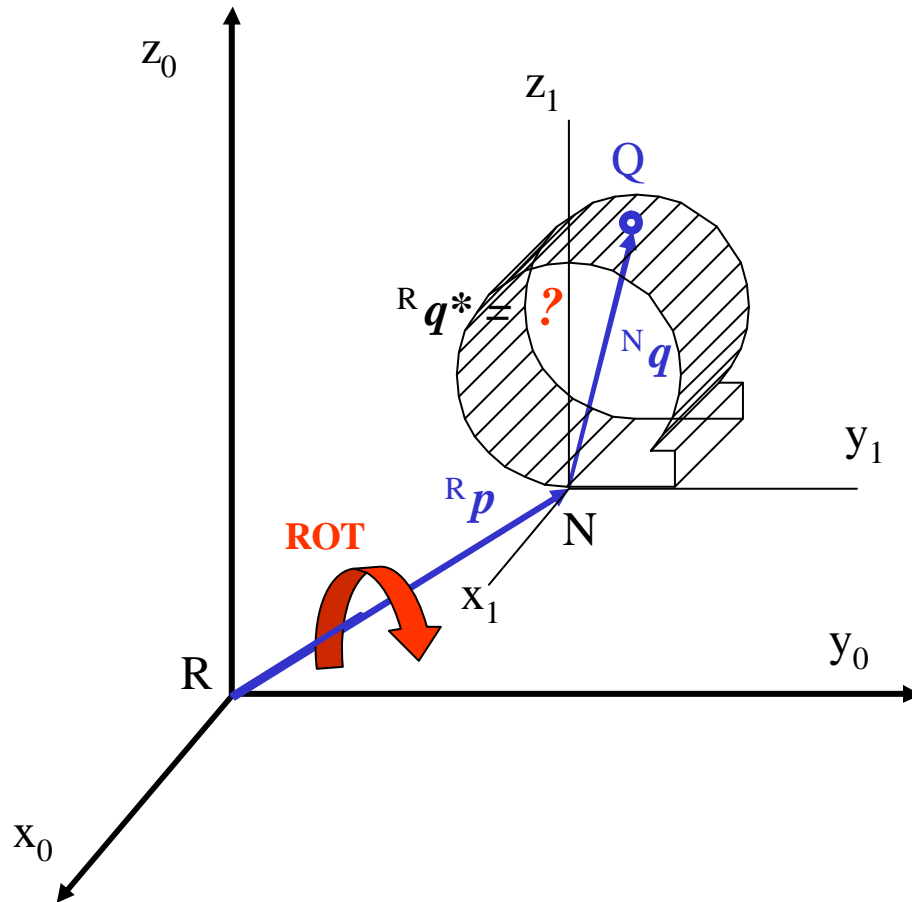


* RECOVERING THE LOCATION OF MOVING OBJECTS: *ROTATION*

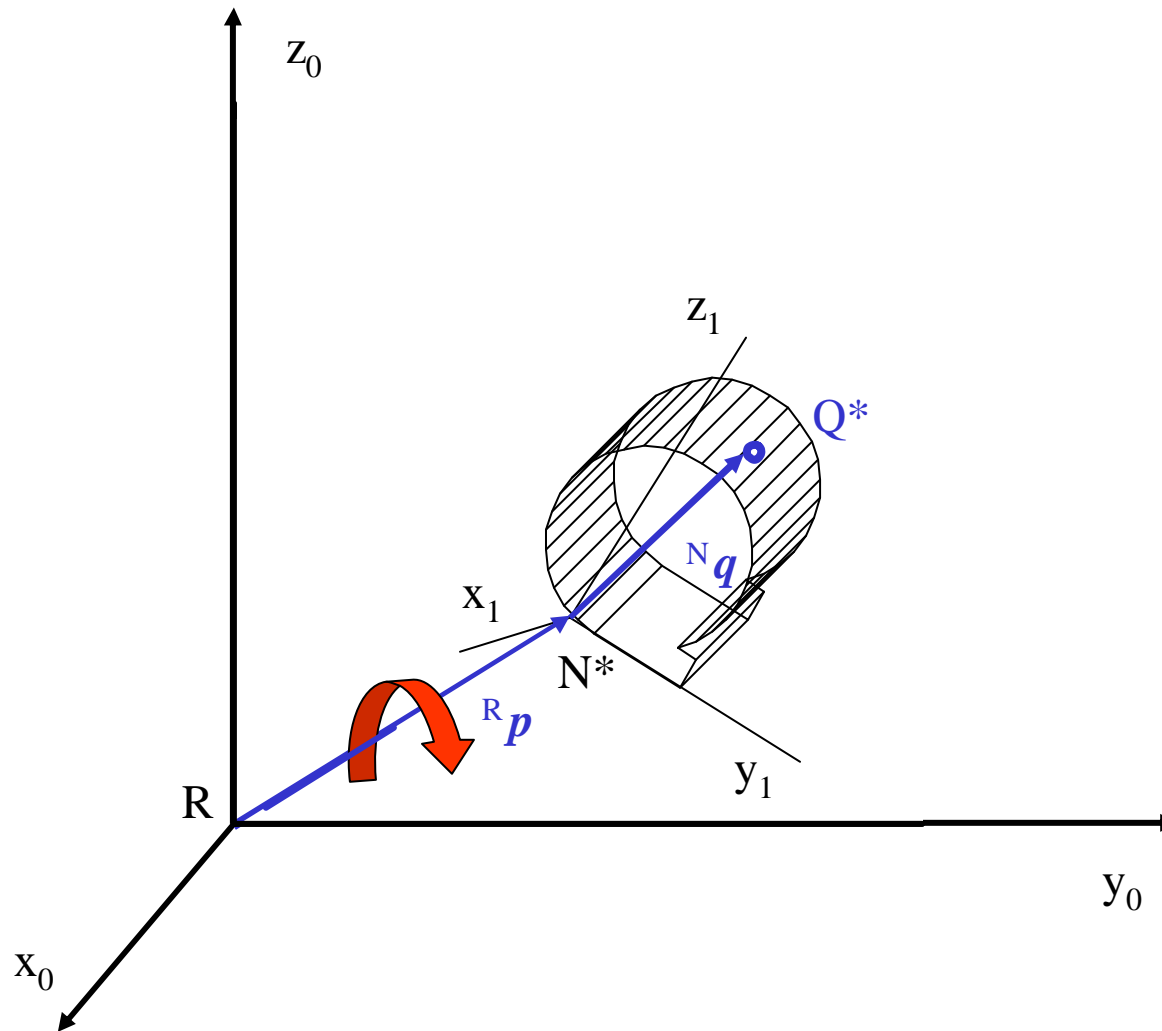
Recovering the 3D location of a point on an object **rotating** in a given reference frame R .

$${}^R q^* = ?$$

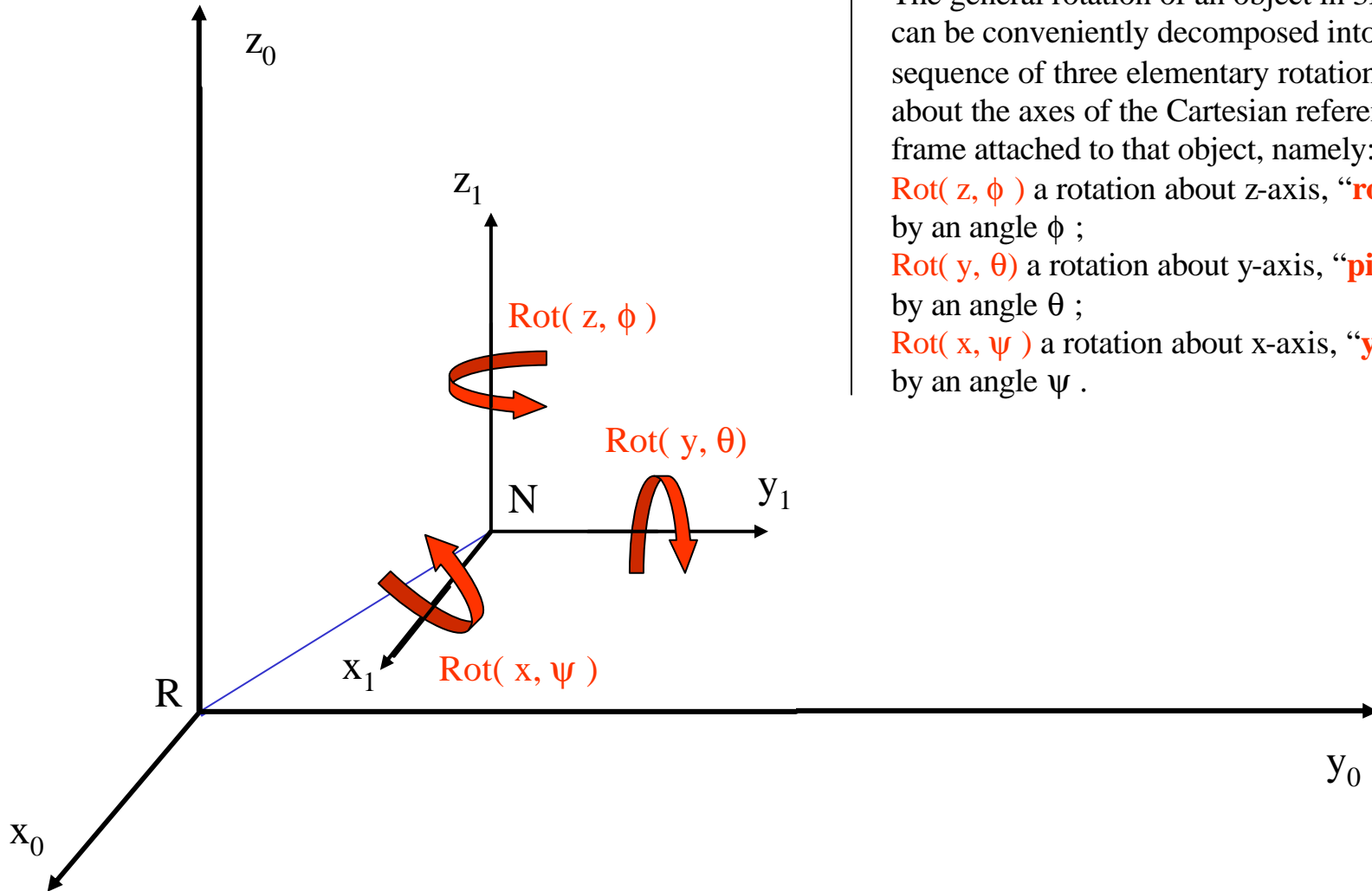
Vectors can not describe the rotation !



RECOVERING THE LOCATION OF MOVING OBJECT: *ROTATION* > continued



RECOVERING THE LOCATION OF MOVING OBJECT: *ROTATION* > continued

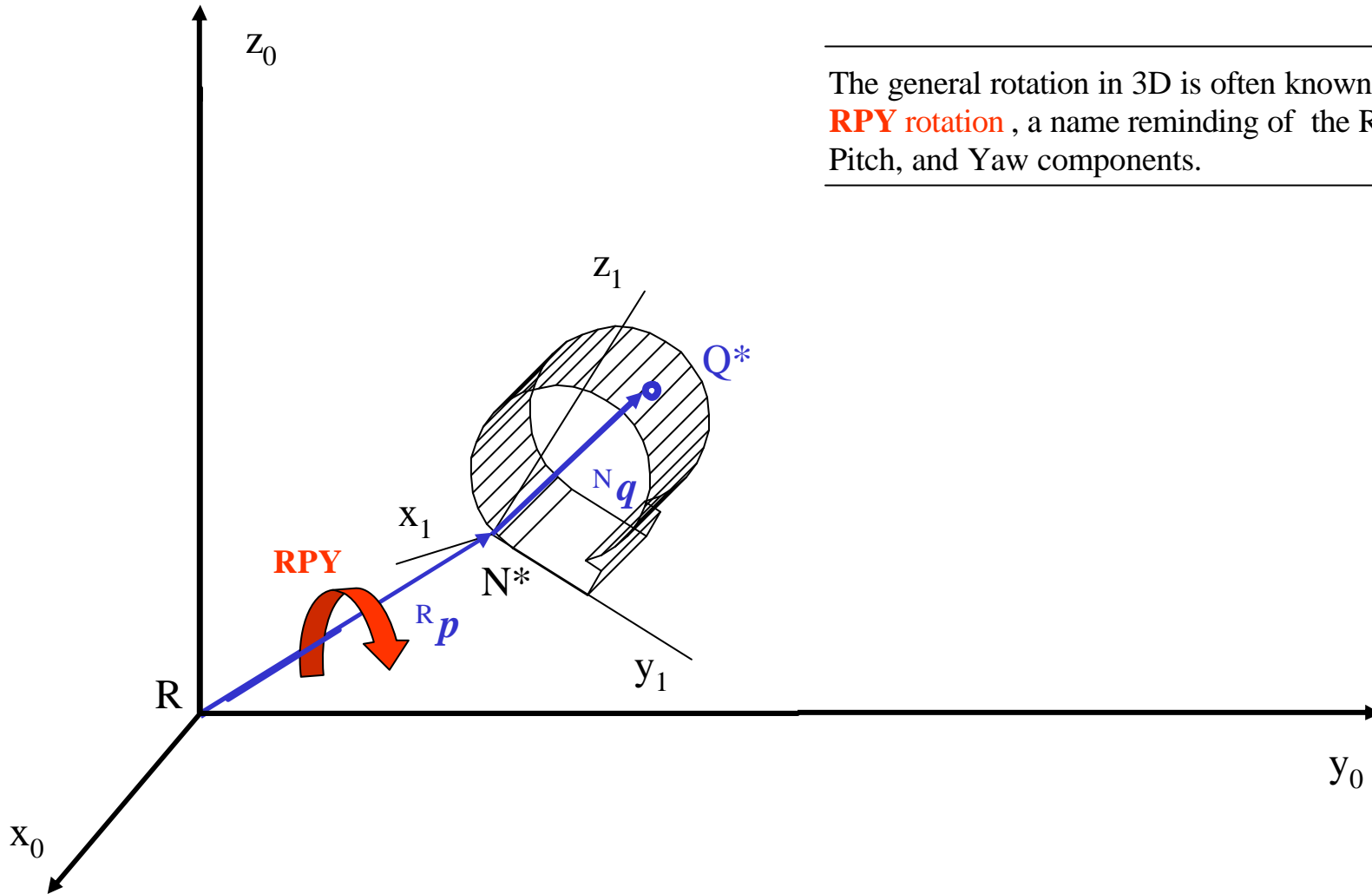


The general rotation of an object in 3D can be conveniently decomposed into a sequence of three elementary rotations about the axes of the Cartesian reference frame attached to that object, namely:

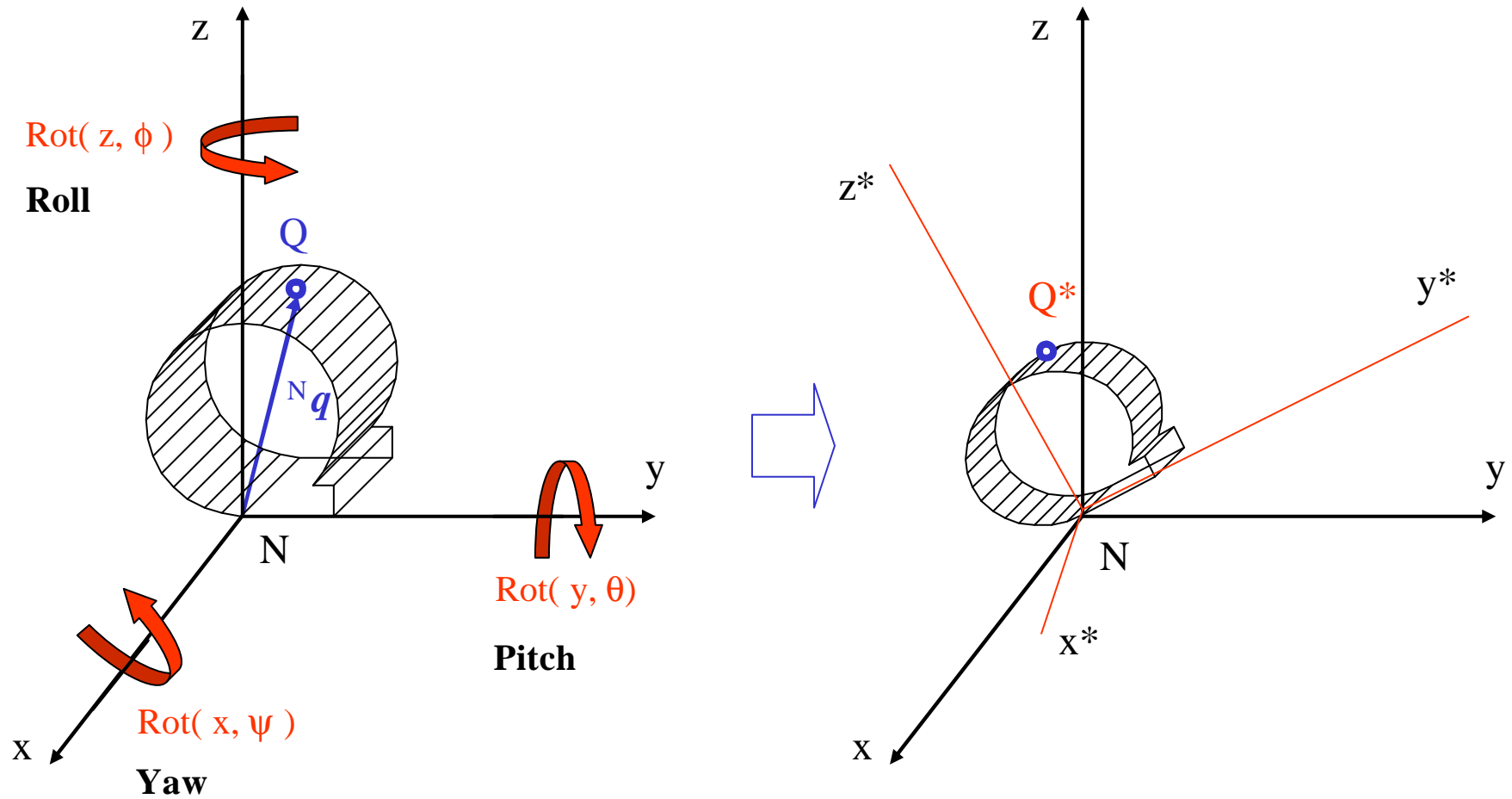
- $Rot(z, \phi)$ a rotation about z-axis, “**roll**,” by an angle ϕ ;
- $Rot(y, \theta)$ a rotation about y-axis, “**pitch**,” by an angle θ ;
- $Rot(x, \psi)$ a rotation about x-axis, “**yaw**,” by an angle ψ .

RECOVERING THE LOCATION OF MOVING OBJECT: *ROTATION* > continued

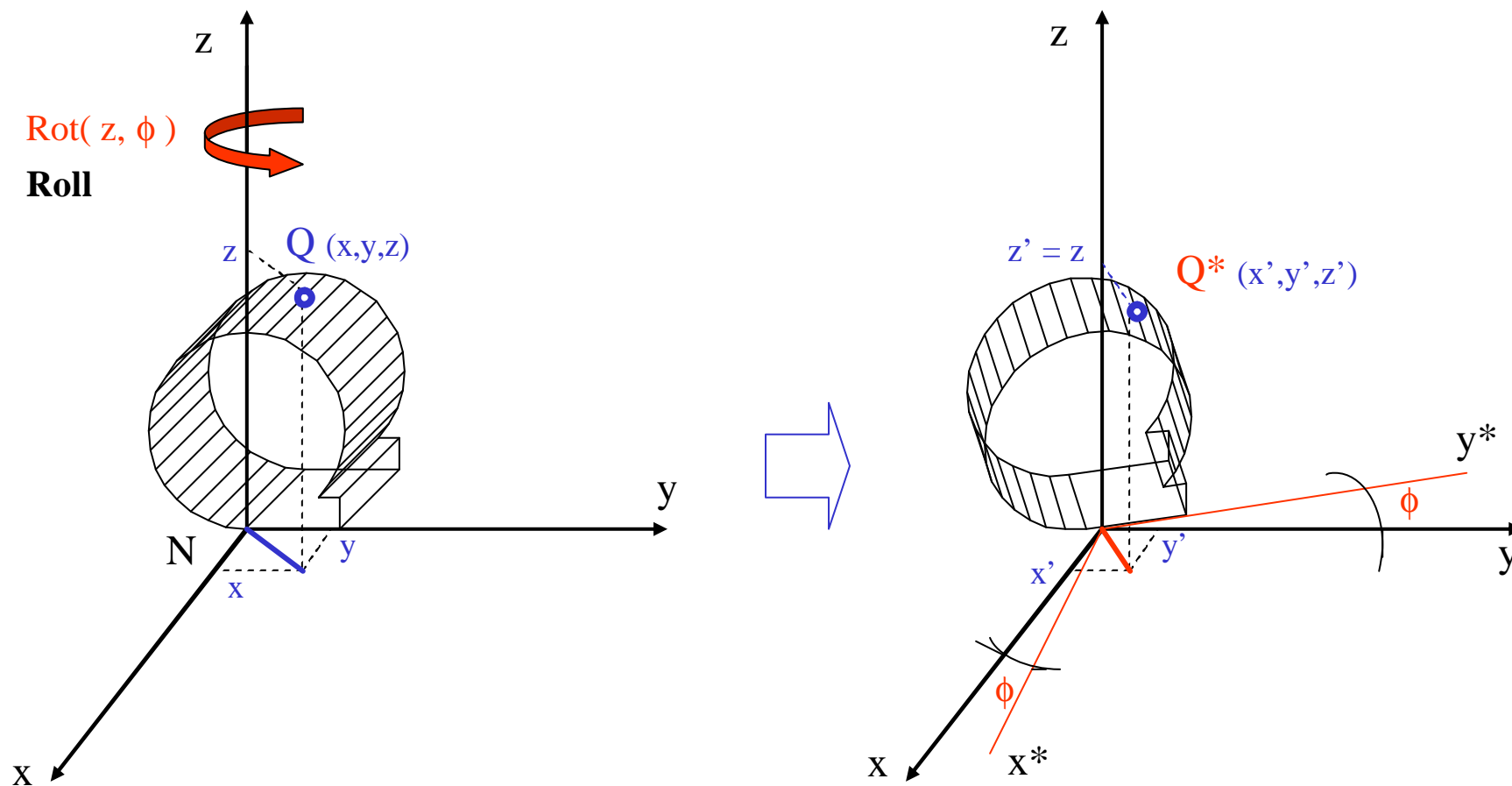
The general rotation in 3D is often known as the **RPY rotation**, a name reminding of the Roll, Pitch, and Yaw components.



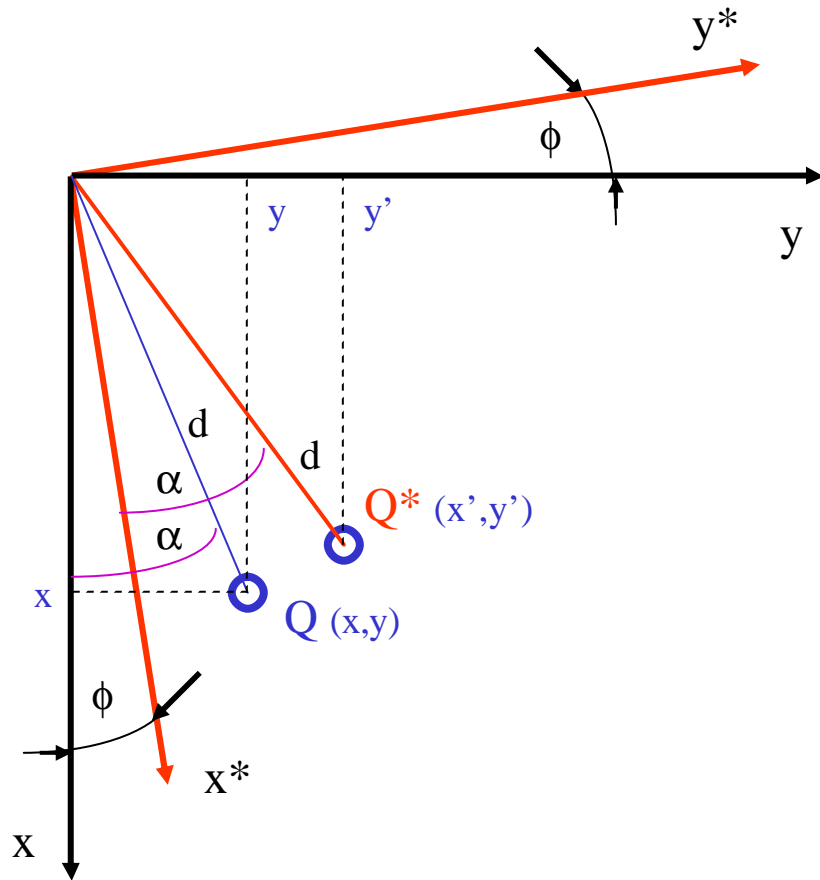
* RECOVERING THE LOCATION OF MOVING OBJECT: ROTATION >> continued



* RECOVERING THE LOCATION OF MOVING OBJECTS: *Roll, Rot(z, f)*



* RECOVERING THE LOCATION OF MOVING OBJECTS: $Rot(z, f)$ > continued



$$y' = d \cdot \sin(\alpha + \phi)$$

$$y' = d \cdot \sin(\alpha) \cdot \cos(\phi) + d \cdot \sin(\phi) \cdot \cos(\alpha)$$

$$x = d \cdot \cos(\alpha)$$

$$y = d \cdot \sin(\alpha)$$

$$y' = y \cdot \cos(\phi) + x \cdot \sin(\phi)$$

In a similar way we get:

$$x' = x \cdot \cos(\phi) - y \cdot \sin(\phi)$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

* 3D HOMOGENEOUS TRANSFORMATION MATRICES FOR ROBOTICS

Homogeneous transformation matrices are general 4-by-4 matrices accounting for both object translation and object rotation in 3D.

$$\mathbf{T} = \left(\begin{array}{ccc|c} & & & \\ & \text{3-by-3} & & \text{3-by-1} \\ & \text{rotation} & & \text{translation} \\ & \text{sub-matrix} & & \text{sub-matrix} \\ \hline & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & & & \mathbf{1} \end{array} \right)$$

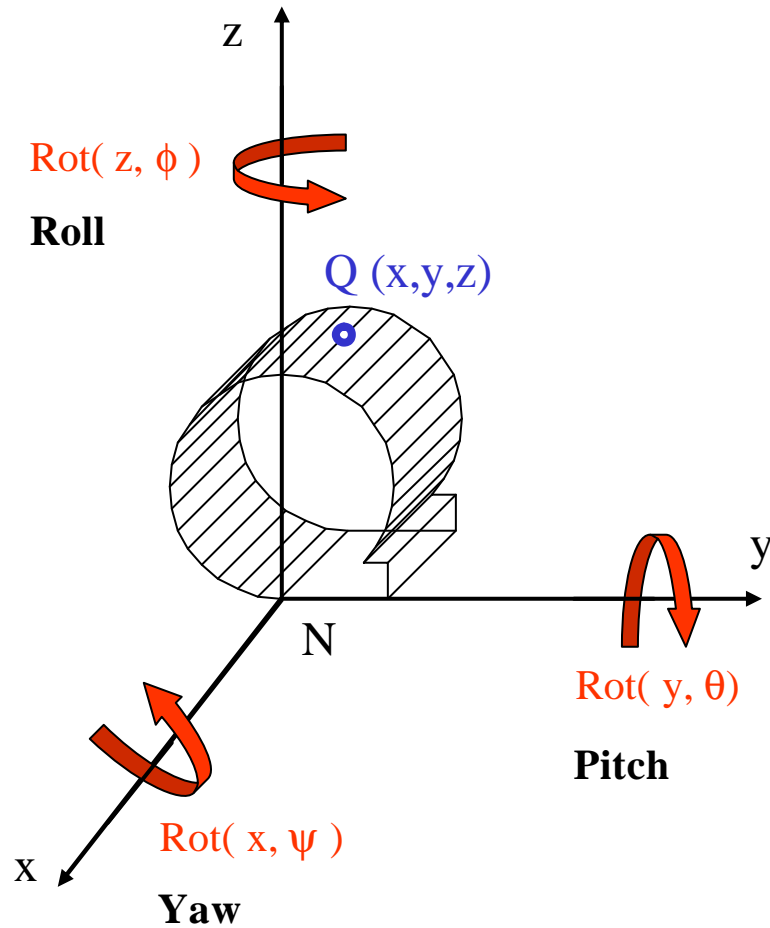


The homogeneous transformations matrix for a general **3D translation** by a vector $p_x \cdot \mathbf{i} + p_y \cdot \mathbf{j} + p_z \cdot \mathbf{k}$ is:

$$\mathbf{Trans}(p_x, p_y, p_z) = \left(\begin{array}{ccc|c} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{array} \right)$$



The homogeneous transformations matrices for **3D rotations** about the x, y, and z axes:



$$\mathbf{Rot}(x, \psi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) & 0 \\ 0 & \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{Rot}(y, \theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{Rot}(z, \phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Homogeneous transformation matrices allow to calculate the final effect of a sequence of object transforms (translations and/or rotations) by multiplying the homogeneous matrices corresponding to these transforms.

