Affine Transformation 2

Computational Visual Design Laboratory
(https://github.com/cvlab) "Roma Tre" Univ, Italy

Computational Graphics 2013

Contents

Introduction

A 3D extension of plane translation and scaling is easy

A major care is only needed for 3D rotation and 3D shearing

In order to unify the tratment of linear and affine transformations, and to use the matrix product as the only geometric operator, we use normalized homogeneous coordinates and tensors in $\mathrm{lin}\ \mathbb{R}^4$

Contents

Translation

Translation tensor $\mathbf{T}_{xyz}(l, m, n)$ with parameters l, m, n (the components of the translation vector), and its matrix:

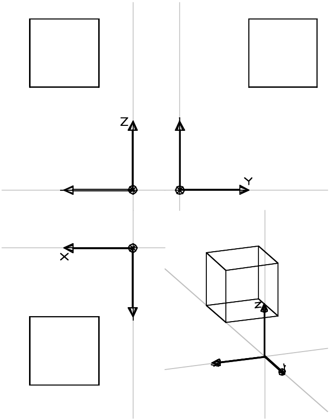
$$\mathbf{T}_{xyz}(l, m, n) = \begin{pmatrix} 1 & 0 & 0 & l \\ 0 & 1 & 0 & m \\ 0 & 0 & 1 & n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Remark

Here the homogeneous row and column are the last ones !!

Translation

Predefined operator in `Pyplasm`




```
T([1,2,3])([0.5,1,1.5])(CUBOID([1,1,1]))
```

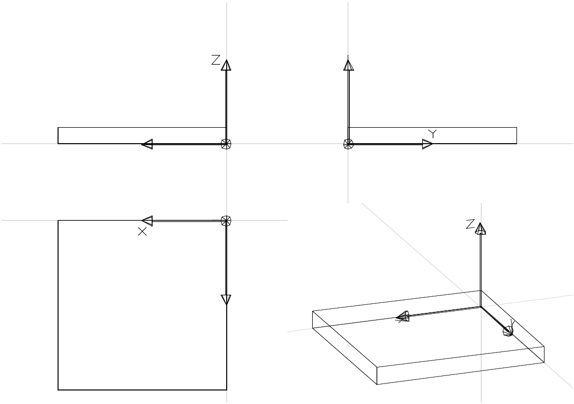
Scaling

The scaling tensor $\mathbf{S}_{xyz}(a, b, c)$ with parameters a, b, c is represented in coordinates by the matrix

$$\mathbf{S}_{xyz}(a, b, c) = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Scaling

Predefined operator in `Pyplasm`




```
S([1,2,3])([2,2,0.2])(CUBOID([1,1,1]))
```

Contents

Elementary rotations

There are $\binom{d}{2}$ different elementary rotations in $\mathbb{E}^d$

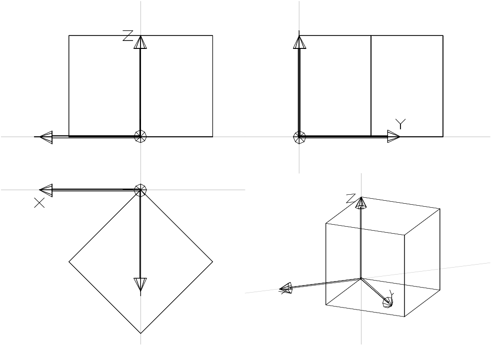
Given a Cartesian frame in $\mathbb{E}^3$, we call elementary rotations $\mathbf{R}_{yz}$,$\mathbf{R}_{xz}$ and $\mathbf{R}_{xy}$, three functions $\mathbb{R} \to \operatorname{lin} \mathbb{R}^3$, which give, for every angle, the rotation tensor about a cooordinate axis

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix}, \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}, \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Matrices in Cartesian coordinates

Elementary rotations

Predefined operator in `Pyplasm`

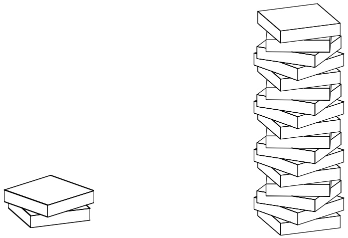



```
R([1,2])(PI/4)(CUBOID([1,1,1]))
```

Elementary rotations

example

Here we define a parallelepiped `element`, translated in x, y by a tensor `T([1,2])([-5,-5])` to move its center upon the z axis



```
element = COMP([T([1,2])([-5,-5]), CUBOID])([10,10,2])
element = T([1,2])([-5,-5])( CUBOID([10,10,2]) )

column = STRUCT( NN(17)([element, T(3)(2), R([1,2])(PI/8)]) )
column = STRUCT( CAT(N(17)([element, T(3)(2), R([1,2])(PI/8)])) )

VIEW(column)
```

equivalent expressions

General rotation

A rotation of $\mathbb{E}^3$ is a linear orthogonal transformation with a set of fixed points (eigenspace in linear algebra) of dimension 1, known as rotation axis

In this transformation, every point (not on the axis) is mapped in the other extreme of a circumference arc of constant angle, centered on the axis, and contained in a plane orthogonal to it.

We will compute the rotation matrix of the tensor $\mathbf{R}_{xyz}(\mathbf{n}, \alpha)$, with

$$\mathbf{R}_{xyz} : \mathbb{R}^3 \times \mathbb{R} \to \operatorname{lin} \mathbb{R}^4 : (\mathbf{n}, \alpha) \mapsto \mathbf{R}_{xyz}(\mathbf{n}, \alpha),$$

where the vector $\mathbf{n}$ is parallel to the rotation axis, and α is the angle of rotation

General rotation

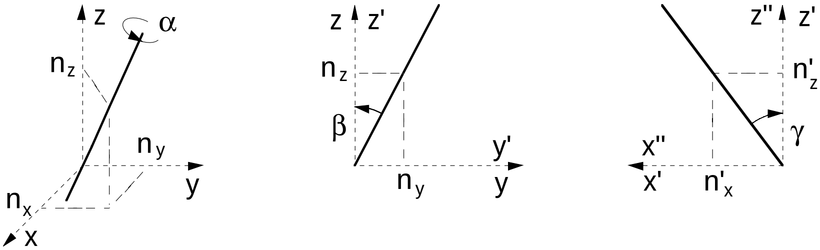
by composition of elementary rotations

A 3D non elementary rotation $\mathbf{R}_{xyz}(\mathbf{n}, \alpha)$, with axis n and α angle, can be reduced to the composition of elementary rotations:

$$
\begin{aligned}
\mathbf{R}_{xyz}(\mathbf{n}, \alpha) &= (\mathbf{R}_y(\gamma) \circ \mathbf{R}_x(\beta))^{-1} \circ \mathbf{R}_z(\alpha) \circ (\mathbf{R}_y(\gamma) \circ \mathbf{R}_x(\beta)) \\
&= \mathbf{R}_x(\beta)^{-1} \circ \mathbf{R}_y(\gamma)^{-1} \circ \mathbf{R}_z(\alpha) \circ \mathbf{R}_y(\gamma) \circ \mathbf{R}_x(\beta) \\
&= \mathbf{R}_x(-\beta) \circ \mathbf{R}_y(-\gamma) \circ \mathbf{R}_z(\alpha) \circ \mathbf{R}_y(\gamma) \circ \mathbf{R}_x(\beta).
\end{aligned}
$$

General rotation

by composition of elementary rotations



(a) **n** axis; (b) rotation about *x*; (c) rotation about *y*

$$\beta = \arctan\left(\frac{n_y}{n_z}\right) \qquad \gamma = -\arctan\left(\frac{n'_x}{n'_z}\right)$$

where $\mathbf{n}' = \mathbf{R}_x(\beta)\,\mathbf{n}$.

General rotation

by transformation of coordinates

The tensor $\mathbf{R}_{xyz}(\mathbf{n}, \alpha)$ of a general rotation may be computed by composition of three tensors:

$$\mathbf{R}_{xyz}(\mathbf{n}, \alpha) = \mathbf{Q}_{\mathbf{n}}^{-1} \circ \mathbf{R}_z(\alpha) \circ \mathbf{Q}_{\mathbf{n}}.$$

such that:

1. a coordinate transformation $\mathbf{Q}_{\mathbf{n}}$ that maps the unit vector $\frac{\mathbf{n}}{|\mathbf{n}|}$ and two orthogonal versors to the elements of a new basis;
2. a rotation $\mathbf{R}_z(\alpha)$ about the z axis of this new basis;
3. the inverse coordinate transformation $\mathbf{Q}_{\mathbf{n}}^{-1}$.

General rotation

by transformation of coordinates

We choose a triple $\mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z$ of orthonormal vectors, with an element oriented as the rotation axis

such vectors are mapped to the basis $\{\mathbf{e}_i\}$ by the unknown matrix $\mathbf{Q_n}$:

$$\begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{pmatrix} = \mathbf{Q_n} \begin{pmatrix} \mathbf{q}_x & \mathbf{q}_y & \mathbf{q}_z \end{pmatrix}.$$

so that

$$\mathbf{Q_n} = \begin{pmatrix} \mathbf{q}_x & \mathbf{q}_y & \mathbf{q}_z \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{q}_x & \mathbf{q}_y & \mathbf{q}_z \end{pmatrix}^T = \begin{pmatrix} \mathbf{q}_x^T \\ \mathbf{q}_y^T \\ \mathbf{q}_z^T \end{pmatrix}$$

General rotation

by transformation of coordinates

Let us start by setting

$$\mathbf{q}_z = \frac{\mathbf{n}}{\|\mathbf{n}\|},$$

We suppose that $\mathbf{n} \neq \mathbf{e}_3$ is verified — if false, it would imply $\mathbf{R}(\mathbf{n}, \alpha) = \mathbf{R}_z(\alpha)$.

Therefore:

$$\mathbf{q}_x = \frac{\mathbf{e}_3 \times \mathbf{n}}{\|\mathbf{e}_3 \times \mathbf{n}\|}, \qquad \text{and} \qquad \mathbf{q}_y = \mathbf{q}_z \times \mathbf{q}_x.$$

General rotation

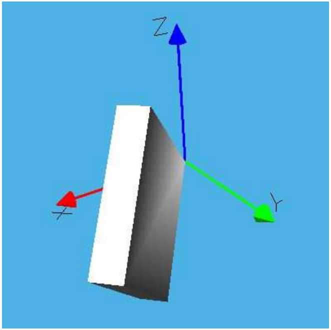
Implementation by transformation of coordinates

```
def ROTN (args):
    alpha, n = args
    n  = UNITVECT(n)
    qx = UNITVECT((VECTPROD([[0,0,1], n])))
    qz = UNITVECT(n)
    qy = VECTPROD([qz,qx])
    Q  = MATHOM([qx, qy, qz])
    if n[0]==0 and n[1]==0:
        return R([1, 2])(alpha)
    else:
        return COMP([MAT(TRANS(Q)),R([1,2])(alpha),MAT(Q)])

VIEW( ROTN([PI/4, [0,0,1]])(CUBE(1)) )
VIEW( ROTN([PI/4, [1,1,1]])(CUBE(1)) )
```

General rotation

example




```
obj = ROTN([ pi/2, [1,1,0] ])(CUBOID([1,1,0.2]))
VIEW(obj)
```

Contents

Elementary shearing

A 3D elementary shearing is a tensor that does'nt change one coordinate of $\mathbb{E}^3$ points, and maps the others as linear functions of the non-transformed coordinate

We may distinguish three elementary shearing tensors $\mathbf{H}_{yz}(a, b)$, $\mathbf{H}_{xz}(a, b)$ and $\mathbf{H}_{xy}(a, b)$, whose matrices differ from the identity only by the elements of a single column

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Elementary shearing

- Let consider the 3D space as a bundle of planes parallel to a coordinate plane, that remains fixed
- The other planes are translated on theirselves, by a linear function of their distance from the fixed plane

$$
\begin{aligned}
\mathbf{p}^* &= \mathbf{H}_x(a, b)\,\mathbf{p} = (x,\ y + ax,\ z + bx,\ 1)^T \\
\mathbf{p}^* &= \mathbf{H}_y(a, b)\,\mathbf{p} = (x + ay,\ y,\ z + by,\ 1)^T \\
\mathbf{p}^* &= \mathbf{H}_z(a, b)\,\mathbf{p} = (x + az,\ y + bz,\ z,\ 1)^T
\end{aligned}
$$

with respect to the tensor $\mathbf{H}_z = \mathbf{H}_{xy}(a, b)$:

1. the $z = 0$ plane is invariant;
2. the $z = 1$ plane translates by the translation vector $\mathbf{t} = (a, b, 0)^T$;
3. each plane $z = c$ translates by a vector $\mathbf{t}' = c(a, b, 0)^T$.