On Computable Coordinate Systems: Definition, Operations, and Computer Code Implementationtion

by **Guojun Pan**

Email: 18858146@qq.com
(Geometric kernel developer in Qingdao, China.)

The author declares that there are no conflicts of interest regarding the publication of this article.

The data used to support the findings of this study have been deposited in the GitHub repository named "Coordinate-System" (https://github.com/panguojun/Coordinate-System).

Abstract: *Matrix operations are often used in coordinate system transformations, while tensor operations are utilized in more general situations. The former belongs to the category of "raw" mathematical objects, which are not designed for coordinate system transformations and have problems with redundancy and convenience in the calculation process. The latter is difficult to master and challenging to use numerically. This paper introduces a specialized structure object for coordinate system transformation as a substitute for matrix and tensor operations.*

1. Introduction

Coordinate systems have always been the most abstract and difficult mathematical and physical objects, and the development of tensor operations has made them even more abstract, requiring years of training to barely master. Therefore, it is necessary to explore a simplified method to address the challenges of coordinate system operations. Although matrix operations are commonly used in coordinate system transformations, they are not designed for this purpose and are too mathematical and vague in meaning. Tensors, on the other hand, are too abstract and difficult to master, and it is difficult for computers to quantify them. A concept specifically designed for coordinate system transformation is needed.

2. Concept

The concept of a coordinate system and the concept of operable coordinates do not come from matrices, but from the use of a ruler. There are two uses for a ruler: 1) Defining a length on the ruler, such as 2 centimeters, and mapping it onto a physical entity for use, commonly used in the process of creating a physical object

from a design.

2) Measuring the length value of a physical entity. This can be extended to general cases, including the measurement of all physical quantities.

Imagine a one-dimensional coordinate system, which corresponds to a ruler. If the ruler uses different units, such as centimeters, and measures a value **v**, to convert it to a universal standard unit, a multiplication is needed:

$$\mathbf{v}0 = \mathbf{v} * cm$$

Here, **v** is the reading on the ruler, and **v**0 is the length value in a universal unit.

In contrast, if we use a ruler to measure the length of a physical entity, it can be expressed as a division operation:

$$\mathbf{v} = \mathbf{v}0 / cm$$

Here, **v** is the reading on the ruler, and **v**0 is the length value in a universal unit.

If we extend this concept to two or three dimensions, this multi-dimensional ruler becomes a coordinate system.

3. Design

A mathematical object called "coord" has been designed using group theory to facilitate arithmetic operations. The structure of the coordinate system is defined as follows: a three-dimensional coordinate system comprises an origin, three directional axes, and three scaling components, which represent translation, rotation, and scaling transformations. We will utilize mathematical expressions that can be overloaded with operators for ease of programming.

a) Definition of coordinate system structure

The coordinate system in three-dimensional space consists of an origin plus three direction axes and three scaling components, corresponding to the three transformations of **translation, rotation, and scaling,** respectively (C++ version):

```
struct coord
{
    vec3 ux, uy, uz;    // Three unit basis vectors
    vec3 s;             // Scaling
    vec3 o;             // Origin Position
}
```

Construct a coordinate system object (C++ version).

By three axes

```
coord C(vec3 ux, vec3 uy, vec3 uz)
coord C(vec3 ux, vec3 uy);
```

By Euler angles

```
coord C(float angle, vec3 axis);
coord C(float pitch, float yaw, float roll);
```

(Origin and scaling can be directly set)

b) Multiplication: Define a vector in a certain coordinate system

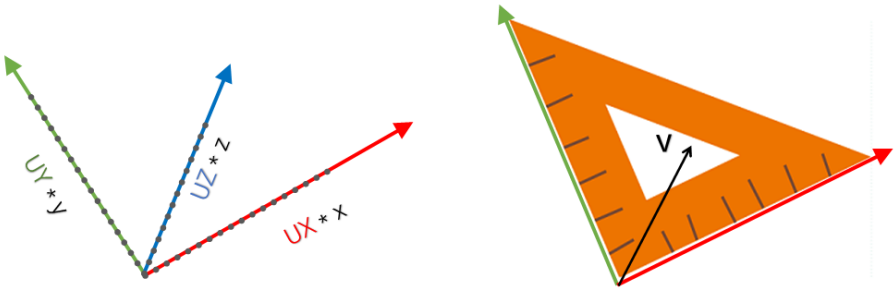


Figure 1: Define a vector in a coordinate system, with each component defined on its respective coordinate axis.

Use multiplication to transform a vector from the local coordinate system to the parent coordinate system and to merge coordinate systems.

For example: V0 is defined in the world coordinate system C0, and V1 is defined in the C1 coordinate system.

$$\mathbf{V}0 = \mathbf{V}1 * C1$$

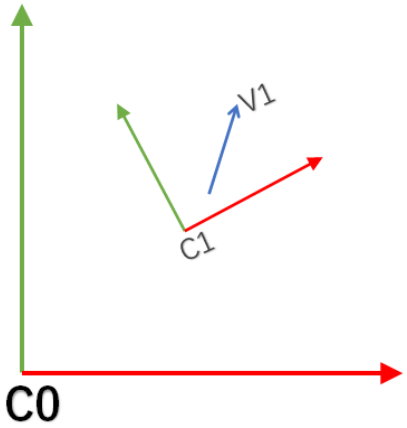


Figure 2: Vector V1 is defined in coordinate system C1, with C1 being the parent coordinate system of C0.

C1, C2, and C3 are local coordinate systems in three scene nodes, and the parent-child direction is:

$$\mathbf{V}0 = \mathbf{V}3 * C3 * C2 * C1 = \mathbf{V}3 * (C3 * C2 * C1)$$

Multiplication after swapping with vectors:

$$\mathbf{V} * C \mathrel{!}= C * \mathbf{V}$$

Define the coordinate system and multiply the vector:

$$C * \mathbf{V} = \mathrm{Lerp}(\mathrm{ZERO}, C, \mathbf{V})$$

Here, Lerp is a linear interpolation function, ZERO is the zero coordinate system.

The multiplication of two vectors can be defined as follows:

$$\mathbf{V}1 * \mathbf{V}2 = \mathbf{ONE} * C1 * \mathbf{ONE} * C2 = \mathbf{ONE} * C1 * C2$$

Here, **ONE** is the unity vector.

c) Division: Measure a vector using a certain coordinate system

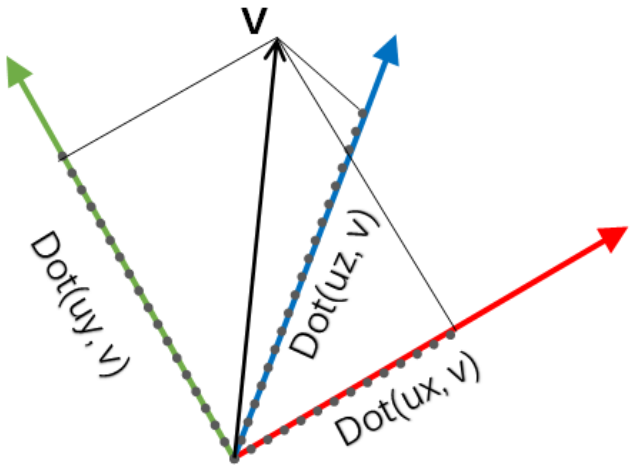


Figure 3: Measuring a vector V using a coordinate system, with each component value being the projection of V onto each coordinate axis.

Use division to project from the parent coordinate system to the local coordinate system. For example:

V0 is defined in the world coordinate system C0, and V1 is defined in the C1 coordinate system:

$$\mathbf{V}1 = \mathbf{V}0 / C1, \mathbf{V}0 = \mathbf{V}1 * C1$$

V2 is defined in the C2 coordinate system:

$$\mathbf{V}2 = \mathbf{V}0 / C2 = \mathbf{V}1 * C1 / C2$$

C1, C2, and C3 are local coordinate systems in three scene nodes, and the parent-child direction is:

$$\mathbf{V}3 = \mathbf{V}0 / C1 / C2 / C3 = \mathbf{V}0 / (C3 * C2 * C1)$$

The division of two vectors can be defined as follows:

$$\mathbf{V}1 / \mathbf{V}2 = \mathbf{ONE} * C1 / (\mathbf{ONE} * C2) = C1 / C2$$

Here, **ONE** is the unity vector.

In addition, due to the lack of multiplication commutativity, the division operation in the coordinate system has two types of division: left division is defined as:

$$C1 \setminus C2 = ONE / C1 * C2.$$

Here, the left division symbol "\" is introduced for comparison with the commonly used right division "/".

However, it is recommended to write the equation in the form on the right side to avoid confusion during calculations.

Of course, the exponential form below can also be used:

$$C1 \setminus C2 = C1^{(-1)} * C2.$$

Here, $C1^{(-1)} = \mathbf{ONE} / C1$

d) Usual Application Scenarios

1. For example, a vector Vw in world space is converted to the local coordinate system C:

   $$\mathbf{V}L = \mathbf{V}w / C$$

   Conversely: $\mathbf{V}w = \mathbf{V}L * C$

2. Conversion between world coordinate system and local coordinate system

   $$C = C3 * C2 * C1$$

   $$\mathbf{V}w = \mathbf{V}L * C, \qquad \mathbf{V}L = \mathbf{V}w / C$$

3. Multiple node hierarchy use

   Vector V5 is defined in the coordinate system of node 5, in the parent node at the level of node 2:

   $$\mathbf{V}2 = \mathbf{V}5 * C5 * C4 * C3$$

   Each coordinate system is a local coordinate system

   Conversely: $\mathbf{V}5 = \mathbf{V}2 / C3 / C4 / C5$

4. Conversion between parallel coordinate systems

   C0{ // C0 under the parent coordinate system

   C1, C2} // Two flat sub-coordinate systems

   Convert a vector from C1 to C2 coordinate systems

$$\mathbf{V}2 = \mathbf{V}1 * C1 / C2$$

e) Addition and subtraction

When calculating the difference between two vectors:

$$\triangle\mathbf{V} = \mathbf{V}1 - \mathbf{V}2$$

Let: $\mathbf{V}1 = \mathbf{V} * C1$, $\mathbf{V}2 = \mathbf{V} * C2$

$$\triangle\mathbf{V} = \mathbf{V} * (C1 - C2)$$

Here defines an addition and subtraction operation for basic addition and subtraction of vectors:

$$\mathbf{V}1 + \mathbf{V}2 = \mathbf{V} * (C1 + C2)$$

$$\mathbf{V}1 - \mathbf{V}2 = \mathbf{V} * (C1 - C2)$$

To facilitate our operations, we establish a principle: for objects of different types, the operation structure remains the same as the object on the left side of the operator.

For example:

$$\mathbf{V}\text{ (vector)} = \mathbf{V}\text{c(vector)} * \text{C(coordinate)}$$

$$\text{C1(coordinate)} = \text{C2 (coordinate)} * \mathbf{V}\text{(vector)}$$

4. Advanced Application Scenarios

a) Coordinate System Differentiation

Constructing a differential coordinate system where a vector can be defined, and the components of the vector correspond to derivatives. This allows for a more intuitive understanding of differentiation operations in space. Additionally, we can perform vector operations in the differential coordinate system, multiplying the resulting vector with the differential coordinate system to obtain a differential vector. This simplifies the expression method.

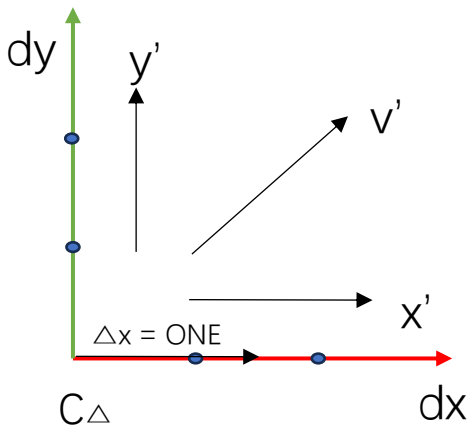


Figure 4: Defining the derivative vector in a differential coordinate system.

Differentiation of the coordinate system in space corresponds to three common forms:

1) Gradient: $\nabla f = \mathbf{U} * df * Cu / (lc * d\mathbf{xyz})$

Here, U represents the principal direction of change, defined in the local coordinate system Cu, df represents the amount of change, Cu

represents the local coordinate system, and d**xyz** represents the vector element defined in the world coordinate system, Ic is the default world coordinate system. The multiplication of Ic and dxyz represents a differential coordinate system.

2) Divergence: $\nabla \cdot \mathbf{F} = d\mathbf{F} / (Ic * d\mathbf{xyz}) \cdot Ic$

Here, d**xyz** represents the vector element defined in the world coordinate system, Ic is the default world coordinate system. The multiplication of Ic and d**xyz** represents a differential coordinate system.

The dot product operation can be appropriately defined between vectors and coordinate systems.

3) Curl: $\nabla \times \mathbf{F} = d\mathbf{F} / (Ic * d\mathbf{xyz}) \times Ic$

Here, d**xyz** represents the vector element defined in the world coordinate system, Ic is the default world coordinate system. The multiplication of Ic and dxyz represents a differential coordinate system.

The cross product operation can be appropriately defined between vectors and coordinate systems.

In order to facilitate numerical calculations, we set a sufficiently small scaling factor eps for the global coordinate system, where we can approximately perform linear operations and ignore higher-order terms. On this scale, we can use the unit one to replace the differential quotient

factor of derivatives, thereby directly using differentials instead of derivatives, achieving the goal of simplifying expressions.

Therefore, the gradient expression mentioned above can be written as:

$$G12 = C1 / C2 - I$$

$$\mathbf{V}12 = \mathbf{V}1 * G12$$

Here C1 and C2 are two adjacent coordinate systems, I is the identity coordinate system. **V**12 is the differential change between the two coordinate systems C1 and C2.

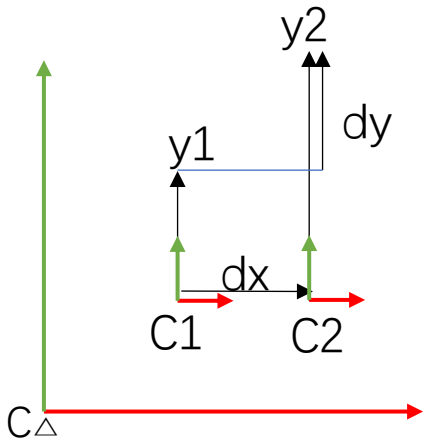


Figure 5: Defining the derivative vector in a differential coordinate system.

In the differential coordinate system, let C△ be the world coordinate system, and C1, C2 be the child coordinate systems. Vectors y1 and y2 are defined as:

$$\mathbf{y}1 = \mathbf{ONE} * C1 \text{ and } \mathbf{y}2 = \mathbf{ONE} * C2$$

The difference between them is given by:

$$\triangle\mathbf{y} = \mathbf{y}2 - \mathbf{y}1 = \mathbf{ONE} * (C2 - C1)$$

Assuming △x represents the positional vector difference from C1 to C2,

$$\triangle\mathbf{x} = \mathbf{ONE} * C12.$$

The derivative is then calculated as:

$$\triangle \mathbf{y} / \triangle \mathbf{x} = \mathbf{ONE} * (C2 - C1) / (\mathbf{ONE} * C12) = \mathbf{ONE} * (C2 - C1) / C12$$

Suppose we need a form:

$$\mathbf{y}1 * G12 = \mathbf{y}2 - \mathbf{y}1 = \mathbf{ONE} * (C2 - C1).$$

Then :

$$G12 = C1 \backslash C2 - I$$

Therefore:

$$\triangle \mathbf{y} / \triangle \mathbf{x} = G12 / C12.$$

If we choose a suitable differential coordinate system such that C12 = I, then:

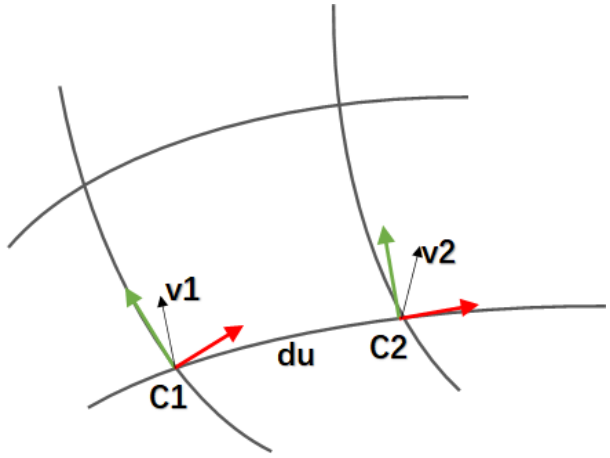
$$\triangle \mathbf{y} / \triangle \mathbf{x} = G12$$

By selecting a suitable differential coordinate system, we can simplify the expression for vector differentiation.

b) Applications in the Field of Differential Geometry

1) Movement of Vectors

Assuming a flat space, a natural coordinate system is defined in which a vector V can move freely without changing. When observed under a curved space coordinate system, V varies at different points, indicating that the coordinate system is position-dependent. Let (1) and (2) be two adjacent points with vectors V1 and V2 at each point, corresponding to coordinate systems C1 and C2. Then, we have:



$$\mathbf{V} = \mathbf{V}1 * C1 = \mathbf{V}2 * C2 \Rightarrow$$

$$\mathbf{V}2 = \mathbf{V}1 * C1 / C2,$$

Let: $R12 = C1 / C2 \Rightarrow$

$$\mathbf{V}2 = \mathbf{V}1 * R12$$

Figure 6: Defining two coordinate systems on a surface space, where vector V1 is translated from one point to another, resulting in V2.

2) Exponential Operation of Coordinate Systems

When a coordinate system is translated along the arc of a differential space, it often involves rotation. To calculate the derivative of rotation with respect to translation, the exponential notation can be utilized. A coordinate system can be represented in exponential form as:

$$C = e^{(\mathbf{n})}$$

$$\mathbf{n} = \mathbf{axis} * \text{angle}$$

where **n** represents the rotation axis multiplied by the angle of rotation.

Utilizing the Taylor series expansion, we can approximate the exponential form as:

$$e^{(\mathbf{n})} \approx 1 + (\mathbf{n}) - .... = \cos(angle) + \sin(angle) * \mathbf{axis}$$

This approximation allows us to calculate the derivative of rotation with respect to translation by replacing division operations in rotation computations with the exponential form.

Considering the coordinate systems at the two ends of a differential space arc, u1 and u2, we have:

$$C1 = e^{(\mathbf{n}1)}, \quad C2 = e^{(\mathbf{n}2)}$$

$$R12 = C1 / C2 = e^{(\mathbf{n}1 - \mathbf{n}2)}$$

let $d\mathbf{n} = \mathbf{n}1 - \mathbf{n}2$, $d\mathbf{u} = \mathbf{u}1 - \mathbf{u}2$:

$$Derivative\ Ru = e^{(d\mathbf{n} / d\mathbf{u})}$$

An aribtrary differential vector can be defined as:

$$d\mathbf{w} = d\mathbf{u} * Ru$$

So to move in any direction of a differential vector:

$$\mathbf{V}2 = \mathbf{V}1 * Ru \wedge (d\mathbf{w} / Ru)$$

This quotient represents the variation of two coordinate system objects C1, C2 in the direction of the unit differential vector **u**.

In addition, the change in two vectors can be computed using the gradient of the coordinate system:

$$\triangle\mathbf{V} = \mathbf{V}2 - \mathbf{V}1 = \mathbf{V}1 * (Gu * (d\mathbf{w} / Ru))$$

where Gu = C1 / C2 - I, I is the unit coordinate system.

5. Curvature Calculation

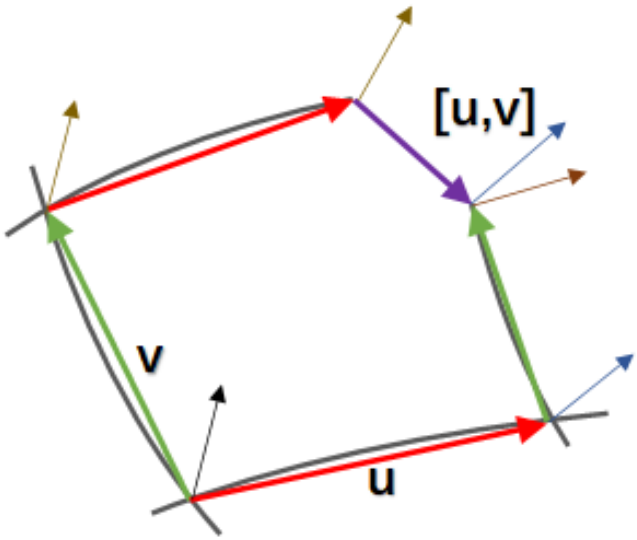


Figure 7: In a small element of the surface space, when a vector is translated along two perpendicular directions, the final result may differ depending on the path taken.

The coordinate system object is capable of converting vectors from the natural coordinate system to the curved coordinate system. The curvature can be calculated by comparing the difference in vector translation along the u->v and v->u paths, using Gu and Gv. Gu and Gv represent the gradient of rotational changes along the u and v vectors. By utilizing the coordinate system, spatial curvature can be calculated, and the coordinate equivalent representation of the Riemann curvature tensor can be given in the u, v coordinate system as:

$$Guv = Gu * Gv - Gv * Gu - G[u,v]$$

Where $Gu = Cu / C0 - I$

$$Gv = Cv / C0 - I$$

I is the identity coordinate system.

C0 corresponds to the initial coordinate system.

Connection vector: [u, v] (Lie bracket operation)

Let **W** = [u, v], **W**u and **W**v are the two components of **W**. Using the coordinate system and vector multiplication mentioned earlier, we can calculate the changes in the gradient coordinate system along them:

$$G[u,v] = Gu*\mathbf{W}u + Gv*\mathbf{W}v$$

6. Combination with Lie groups, Lie algebras

The rotation matrix R, which describes the rotation of an object in three-dimensional space, can be classified as a special type of matrix belonging to a mathematical structure known as a Lie group. A Lie group is a mathematical object that possesses both continuous and group structure, and it finds extensive applications in physics and geometry.

In a similar vein, our coordinate system C can be considered as an element of the Lie group. This implies that we can use the multiplication operation of the Lie group to represent transformations between coordinate systems. Specifically, the multiplication operation of our coordinate system can be expressed as matrix multiplication, with the identity element ONE representing the identity transformation.

Hence, we can represent our coordinate system as:

$$C \in \{ONE, (*)\}$$

where (*) denotes the multiplication operation of the coordinate system.

Utilizing the concept of Lie groups, we can derive the expression for the vector cross product in terms of coordinate systems. Let **v**1 and **v**2 be two vectors, and C1 and C2 be the corresponding elements of the coordinate system. The cross product of **v**1 and **v**2 can be written as:

$$\mathbf{v}1 \times \mathbf{v}2 = \mathbf{v} * C1 \times (\mathbf{v} * C2)$$

Here, **v** represents the coordinate system representation of vectors **v**1 and **v**2, which can be expressed as the unity vector **ONE**. Simplifying the equation, we obtain:

$$\mathbf{v}1 \times \mathbf{v}2 = \mathbf{ONE} * (C1 \times C2)$$

To further describe the cross product operation of coordinate systems, we introduce the concept of Lie algebra. We define the coordinate system cross product operation using Lie algebra brackets:

$$[C1, C2] = C1 * C2 - C2 * C1$$

This expression represents the operation of multiplying C1 and C2 and subtracting the result from the multiplication of C2 and C1, as defined by Lie algebra. This abstract representation provides a concise way to describe the operations between coordinate systems.

7. Conclusion

In conclusion, the use of coordinate system objects offers a simplified approach to linear operations in local space. By appropriately designing

operations for vector movement and rotation, we can effectively streamline complex calculations. Moreover, by adhering to the principles of group theory, a unified form can be achieved, enhancing the efficiency and coherence of the overall process.

Coordinate system objects find extensive application in scenarios that involve numerous linear operations, such as constraint calculations. By utilizing these objects, we can effectively manage and manipulate vectors, matrices, and tensors, seamlessly integrating them into a cohesive framework. This integration extends to the unification of Lie groups and Lie algebraic objects, which are combined into a cohesive ring structure.

The utilization of coordinate system objects, in conjunction with Lie groups and Lie algebra, allows for a comprehensive and unified approach to mathematical operations. This unified framework not only simplifies complex calculations but also provides a consistent and coherent methodology for handling various mathematical entities. Overall, the integration of coordinate system objects with Lie groups and Lie algebraic objects offers a powerful and versatile tool for solving problems that involve extensive linear operations.

8. Partial code implementation

Implement using C++:

```
vec3 operator * (const vec3& p, const coord3& c)
{
	return c.ux * (c.s.x * p.x) + c.uy * (c.s.y * p.y) + c.uz * (c.s.z * p.z) + c.o;
}
coord3 operator * (const coord3& c1, const coord3& c2)
{
```

```
	coord3 rc;
	rc.ux = c1.ux.x * c2.ux + c1.ux.y * c2.uy + c1.ux.z * c2.uz;
	rc.uy = c1.uy.x * c2.ux + c1.uy.y * c2.uy + c1.uy.z * c2.uz;
	rc.uz = c1.uz.x * c2.ux + c1.uz.y * c2.uy + c1.uz.z * c2.uz;
	rc.s = s * c.s;
	rc.o = c1.o.x * c2.s.x * c2.ux + c1.o.y * c2.s.y * c2.uy + c1.o.z * c2.s.z * c2.uz + c2.o;
	return rc;
}
vec3 operator / (const vec3& p, const coord3& c)
{
	vec3 v = p - c.o;
	return vec3( v.dot(c.ux) / c.s.x, v.dot(c.uy) / c.s.y,    v.dot(c.uz) / c.s.z );
}
coord3 operator / (const coord3& c1, const coord3& c2)
{
	coord3 rc;
	rc.ux = vec3(c1.ux.dot(c2.ux), c1.ux.dot(c2.uy), c1.ux.dot(c2.uz));
	rc.uy = vec3(c1.uy.dot(c2.ux), c1.uy.dot(c2.uy), c1.uy.dot(c2.uz));
	rc.uz = vec3(c1.uz.dot(c2.ux), c1.uz.dot(c2.uy), c1.uz.dot(c2.uz));
	rc.s = c1.s / c2.s;
	rc.o = c1.o - c2.o;
	rc.o = vec3(rc.o.dot(c2.ux) / c2.s.x, rc.o.dot(c2.uy) / c2.s.y, rc.o.dot(c2.uz) / c2.s.z);
	return rc;
}
coord3 operator + (const coord3& c1, const coord3& c2)
{
	coord3 rc;
	rc.ux = c1.VX() + c2.VX();
	rc.uy = c1.VY() + c2.VY();
	rc.uz = c1.VZ() + c2.VZ();
	rc.norm();
	rc.o = o + c2.o;
	return rc;
}
coord3 operator - (const coord3& c1, const coord3& c2)
{
	coord3 rc;
	rc.ux = c1.VX() - c2.VX();
	rc.uy = c1.VY() - c2.VY();
	rc.uz = c1.VZ() - c2.VZ();
	rc.norm();
	rc.o = o - c2.o;
	return rc;
}
```

Reference:

1. Hamilton, W. R. (1847). On Quaternions. Proceedings of the Royal Irish Academy, 3, 1-16.

2.Koepf, W. T., & Salzmann, H. R. (2003). Curvature Tensor and Curvature Fo rms for Riemannian and Lorentzian Manifolds. Journal of Differential Geome try, 65(3), 457-499.

3. Hall, Brian C. "Lie groups, Lie algebras, and representations: an elementary introduction." Springer Science & Business Media, 2003.

Hall, Brian C. "Lie groups, Lie algebras, and representations: an elementary introduction." Springer Science & Business Media, 2003. (This book provides a comprehensive introduction to Lie groups, Lie algebras, and their representations.)

4. Varadarajan, V. S. "Lie groups, Lie algebras, and their representations." Springer Science & Business Media, 1984. (This book covers the theory of Lie groups and their representations in a detailed manner.)

5. Fulton, William, and Joe Harris. "Representation theory: a first course." Springer Science & Business Media, 2013.

6. Humphreys, James E. "Introduction to Lie algebras and representation theory." Springer Science & Business Media, 1978.

7. Lee, John M. "Introduction to smooth manifolds." Springer Science & Business Media, 2012.

8. Do Carmo, Manfredo Perdigão. "Riemannian geometry." Birkhäuser, 1992.

9. Kobayashi, Shoshichi, and Katsumi Nomizu. "Foundations of differential geometry." Vol. 1. Wiley, 1996.