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

by **Guojun Pan**

（Geometric kernel developer in Qingdao, China.）

The author(s) declare(s) that there are no conflicts of interest regarding the publication of this article.

***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

2. The 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:

$$v0 = v * cm$$

Here, v is the reading on the ruler, and V0 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:

$$v = v0 / cm$$

Here, v is the reading on the ruler, and V0 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 named "coord" is designed based on group theory to perform arithmetic operations. The definition of the coordinate system structure is as follows: a three-dimensional coordinate system consists of an origin, three directional axes, and three scaling components, representing translation, rotation, and scaling transformations.

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

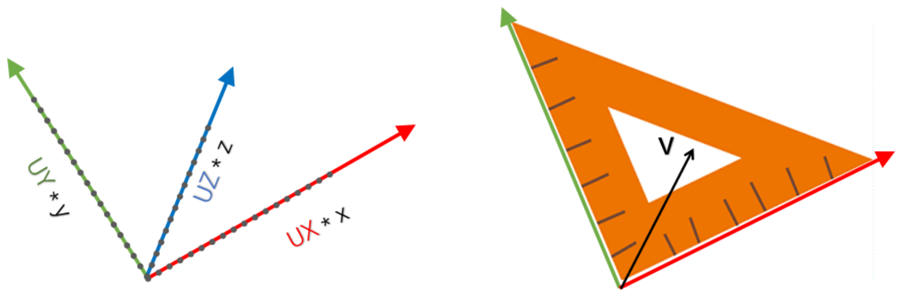

Figure 1: Define a vector in a coordinate system, with each component defined on its respective coordinate axis.

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.

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

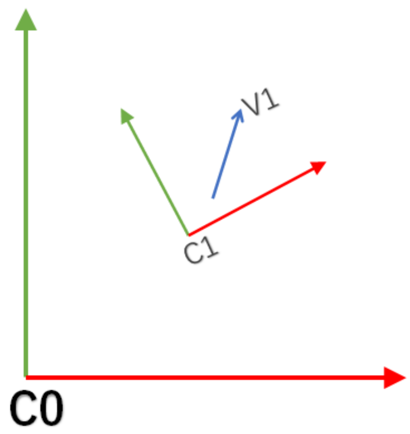

Figure 2: Vector V1 is defined in coordinate system C1, with C1 being the parent coordinate system of C0.

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

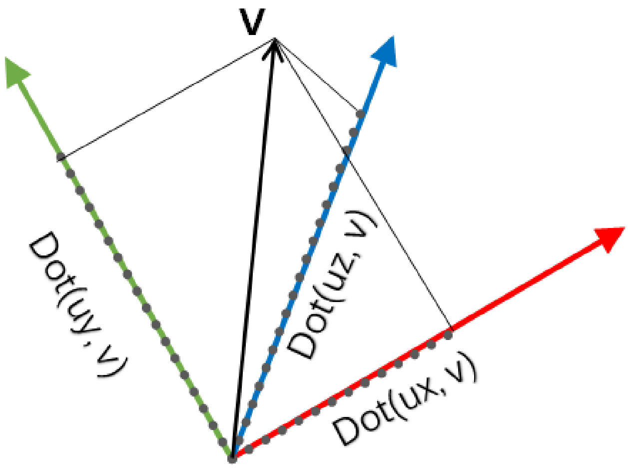

Figure 3: Measuring a vector V using a coordinate system, with each component value being the projection of V onto each coordinate axis.

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:

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

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

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: $\mathbf{V}3 = \mathbf{V}2 / C3 / C4 / C5$

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:

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

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

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

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:

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

C1(coordinate) = C2 (coordinate) * $\mathbf{V}$(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. In the following description, I will adopt this approach.

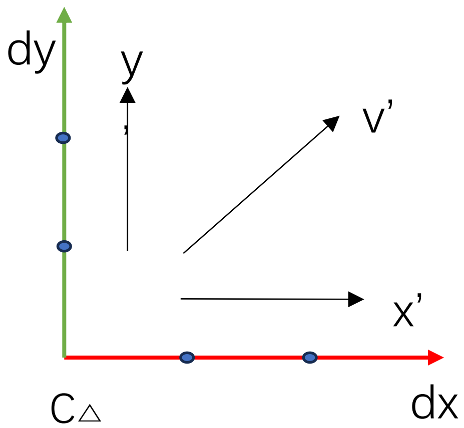

Figure 4: Defining the derivative vector in a differential coordinate system.

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

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

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 dxyz 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:

$$G = C2 / C1 - I$$

$$d\mathbf{V} = \mathbf{V}1 * G$$

Here C1 and C2 are two adjacent coordinate systems that are a unit length apart. I is the identity coordinate system. d**V** is the differential change between the two coordinate systems C1 and C2.

b) Applications in the Field of Differential Geometry

5. Curvature Calculation

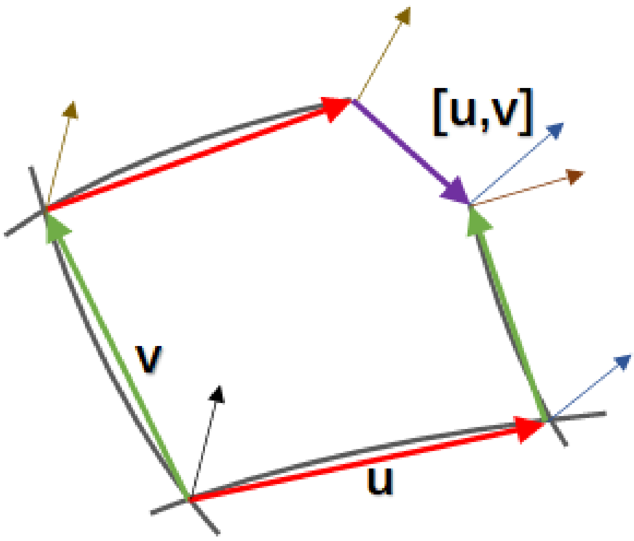

Figure 6: In a small element of the surface space, when a vector is translated along two perpendicular directions, the final result may differ depending on the path taken.

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 exponential 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 belongs to a Lie group, and the multiplication operation of our coordinate system is equivalent to the rotation matrix. Therefore, our coordinate system C is also a Lie group element, with multiplication as its operation and ONE as its identity element:

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

The vector fork multiplies the corresponding coordinate system

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

Let v = vec3. ONE, then C * v = C, so the above equation can be written as:

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

We define a coordinate system cross product operation expressed in Lie algebra brackets:

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

7. Conclusion

The coordinate system object can simplify the linear operation in local space, and some operations can be appropriately designed when calculating vector movement and rotation operations, and a unified form can be formed according to the rules of group theory. It can be applied to situations that require a lot of linear operations, such as constraint calculations. Vectors, matrices, and tensors are combined through coordinate system objects, and Lie groups and Lie algebraic objects are unified into a ring, forming a unified and coherent form.

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 Forms for Riemannian and Lorentzian Manifolds. Journal of Differential Geometry, 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.