A Mathematical Object for Coordinate System Transformation

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 "element" 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. 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

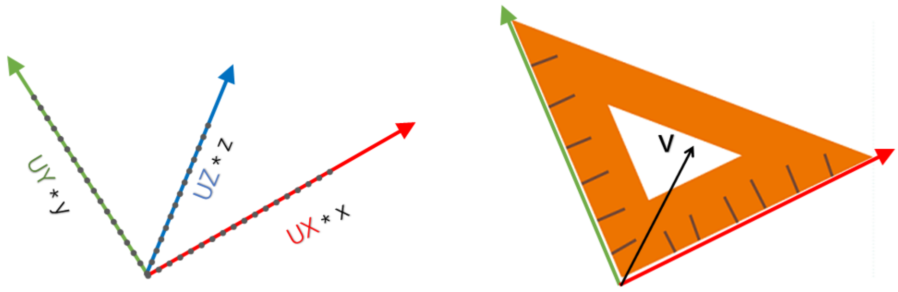


Figure 1: Define a vector in a coordinate system, with each component defined on its respective coordinate axis.

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.

$$V0 = V1 * C1$$

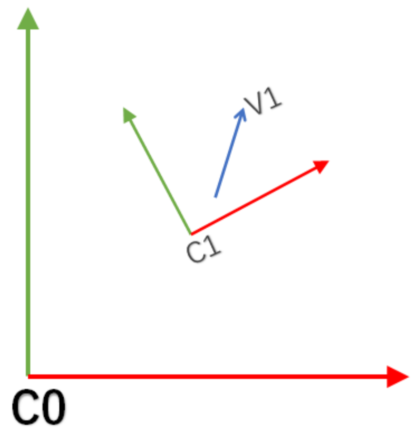


Figure 2: Vector V1 is defined in coordinate system C1, with C1 being the parent coordinate system of C0.

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

$$V0 = V3 * C3 * C2 * C1 = V3 * (C3 * C2 * C1)$$

Multiplication after swapping with vectors:

V * C != C * V

Define the coordinate system and multiply the vector:

C * V = Lerp(ONE, C, V.dot(UX))

where Lerp is a linear interpolation function and

ONE = coord. ONE, UX = vec3. UX

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

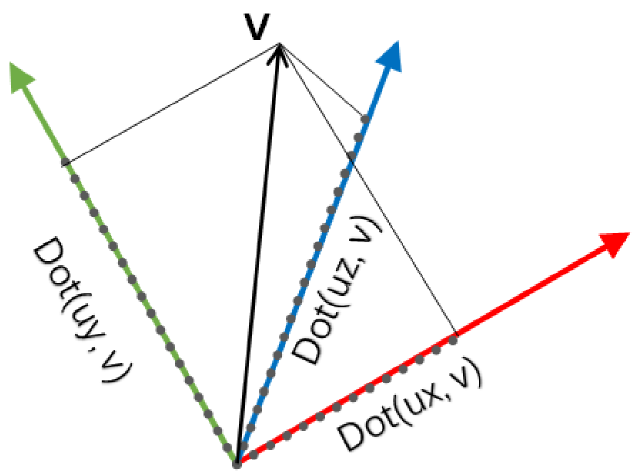


Figure 3: Measuring a vector V using a coordinate system, with each component value being the projection of V onto each coordinate axis.

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

V1 = V0 / C1, V0 = V1 * C1

V2 is defined in the C2 coordinate system:

V2 = V0 / C2 = V1 * C1 / C2

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

$$V3 = V0 / C1 / C2 / C3 = V0 / (C1 * C2 * C3)$$

d) Usual Application Scenarios

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

$$VL = Vw / C$$

Conversely: $Vw = VL * C$

2. Conversion between world coordinate system and local coordinate system

$$C = C3 * C2 * C1$$

$$Vw = VL * C, \qquad VL = Vw / C$$

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:

$$V2 = V5 * C5 * C4 * C3$$

Each coordinate system is a local coordinate system

Conversely: $V3 = V2 / C3 / C4 / C5$

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

$V2 = V1 * C1 / C2$

e) Addition and subtraction

When calculating the difference between two vectors:

$DV = V1 – V2$

Let: $V1 = v * C1$, $V2 = v * C2$

$DV = v * (C1 – C2)$

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

$V1 + V2 = v * (C1+ C2)$

$V1 - V2 = v * (C1- C2)$

3. Advanced Application Scenarios

a) Coordinate System Differentiation

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

1) Gradient: $\nabla f = dF(U * df * Cuvw) / dxyz$

Here, U represents the basis vector U in the local coordinate system, df represents the amount of change, Cuvw represents the local coordinate system, and dxyz represents the vector element defined in the world coordinate system.

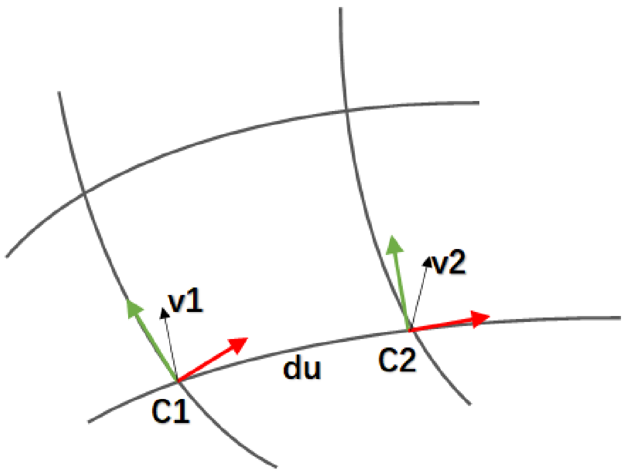
2) Divergence: $\nabla \cdot F = dF / dxyz \cdot Ic$

3) Curl: $\nabla \times F = dF / dxyz \times Ic$

b) Applications in the Field of Differential Geometry

1) Movement of Vectors

Assuming a straight space, a natural coordinate system is defined under V where the vector 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:



$V = V1 * C1 = V2 * C2 \Rightarrow$

$V2 = V1 * C1 / C2,$

Let: $G12 = C1 / C2 \Rightarrow$

$V2 = V1 * G12$

Figure 4: Defining two coordinate systems on a surface space, where vector V1 is translated from one point to another, resulting in V2.

To move in any direction of the differential vector:

Arbitrary differential vector:

$dW = dU * Cw$

So move in any direction of the differential vector:

$V2 = V1 * Gw = V1 * Gu * Cw$

where: $Gu = G12 = C1 / C2$

This quotient represents two coordinate system objects in the direction of the unit differential vector dU.

2) Curvature calculation

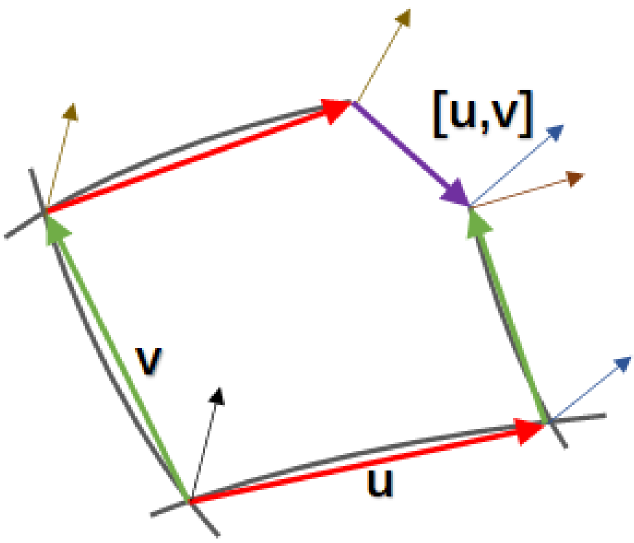


Figure 5: In a small element of the surface space, when a vector is translated along two perpendicular directions, the final result may differ depending on the path taken.

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:

Ruv = Gu * Gv - Gv * Gu - G [uv]

Where Gu = UG - ONE

UG = C2 / C1

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

4. 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

$$v1 \times v2 = v * C1 \times (v * C2)$$

Let v = vec3. ONE is vec3(1,0,0), then C * v = C, so the above equation can be written as:

$$v1 \times v2 = v * (C1 \times C2)$$

We define a coordinate system fork multiplication operation expressed in Lie algebra brackets:

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

5. 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.