球谐函数及其好处

在谈及球谐函数时，从傅里叶级数开始理解，会有非常好的体验。从球谐函数的定义出发：球面函数可以分解为无限个球谐函数的累加。阶数越高，逼近的效果越好。而傅里叶级数的定义是，任意的周期函数都可以由简单震荡的无限叠加获得。当然，我们在实际计算机中都是用有限个数去逼近，也就是离散化。

和傅里叶级数性质类似，球谐函数也是以正交函数作为基底，傅里叶级数的正交基底为sin(nx)和cos(nx)。球谐函数则是球面上的正交基底。其主要性质有：

- 标准正交性
- 旋转不变性
- 函数乘积的积分等于其球谐系数向量的点积

公式

标准正交性表示：

$$\int y_i y_j = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$

其中 y_i, y_j 表示两组球谐函数基底。

旋转不变性表示：

$$g(t) = f(R(t)) = R(f(t))$$

函数乘积的积分等于其球谐系数向量的点积：

$$\int f(t)g(t)dt = \sum_{i=0}^{n^2} F_i G_i$$

我们列出球谐函数的表达式：

$$Y_l^m(\theta, \phi) = \begin{cases} \sqrt{2K_l^m}\cos(m\phi)P_l^m(\cos\theta) & (m > 0) \\ \sqrt{2K_l^m}\sin(-m\phi)P_l^{-m}(\cos\theta) & (m < 0) \\ K_l^0 P_l^0(\cos\theta) & (m = 0) \end{cases}$$

我们可以把它理解为加上正版的傅里叶级数正交基底。其中 K_l^m 为：

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}$$

我们可以根据公式写出求解 K_l^m 的代码：

```
double Harmonics::K(const int l, const int m)
{
    //m小于0的情况在外部乘上-1传入
    return (sqrt(((2 * l + 1) * Factorial(l - m)) / (4 * PI * Factorial(l +
m))));
}

//求解一次阶乘
int Harmonics::Factorial(int v)
```

```
{
    if (v == 0)
        return (1);

    int result = v;
    while (--v > 0)
        result *= v;
    return (result);
}
```

P_l^m为伴随勒让德多项式 (ALP)。首先勒让德函数 $P(x)$ 是勒让德微分方程的解：

$$(1-x^2)\frac{d^2P(x)}{dx^2} - 2x\frac{dP(x)}{dx} + n(n+1)P(x) = 0$$

勒让德多项式表达式为：

$$P_n(x) = \frac{1}{2^n \cdot n!}\frac{d^n}{dx^n}[(x^2-1)^n]$$

伴随勒让德多项式 (ALP) 引入了两个参数 l 和 m，m 来自于勒让德多项式定义，其表达式为：

$$P_l^m(x) = (-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}(P_l(x))$$

我们求解 ALP 时，可以使用三条递归公式：

$$(l-m)P_l^m = x(2l-1)P_{l-1}^m - (l+m-1)P_{l-2}^m$$

$$P_m^m = (-1)^m(2m-1)!!(1-x^2)^{m/2}$$

$$P_{m+1}^m = x(2m+1)P_m^m$$

通过这三条递归公式，可以求解任意阶的伴随勒让德多项式。我们根据公式写出如下代码：

```
double Harmonics::P(const int l, const int m, const double x)
{
    // 公式二不需要递归计算
    if (l == m)
        return (pow(-1.0f, m) * DoubleFactorial(2 * m - 1) * pow(sqrt(1 - x * x),
m));

    // 公式三
    if (l == m + 1)
        return (x * (2 * m + 1) * P(m, m, x));

    // 公式一
    return ((x * (2 * l - 1) * P(l - 1, m, x) - (l + m - 1) * P(l - 2, m, x)) /
(l - m));
}

//二次阶乘
int Harmonics::DoubleFactorial(int x)
{
    if (x == 0 || x == -1)
        return (1);

    int result = x;
```

```
    while ((x -= 2) > 0)
        result *= x;
    return (result);
}
```

完成P_l^m和K_l^m的计算后，我们可以根据表达式计算球谐函数了，下面给出代码：

```
double Harmonics::y(const int l, const int m, const double theta, const double
phi)
{
    if (m == 0)
        return (K(l, 0) * P(l, 0, cos(theta)));

    if (m > 0)
        return (sqrt(2.0f) * K(l, m) * cos(m * phi) * P(l, m, cos(theta)));

    // 当m小于0时，预先乘上-1，再传给K
    return (sqrt(2.0f) * K(l, -m) * sin(-m * phi) * P(l, -m, cos(theta)));
}
```

$f(\theta,\phi)$ 在球面的展开式为：

$$f(\theta,\phi)=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}C_l^mY_l^m(\theta,\phi)$$

其中 C_l^m 是球谐系数，Y_l^m 为球谐函数。这个公式和傅里叶级数是非常相似的。同样，我们求解系数 C_l^m 的过程也十分相似：

$$C_l^m=\int_S f(s)Y_l^m(s)ds$$

其中 $f(s)$ 为原函数。当我们求解完成相应阶数的系数后，可以重构 $f(s)$，我们重构后的函数为：

$$\hat{f}(s)=\sum_{l=0}^{n-1}\sum_{m=-l}^{l}C_l^mY_l^m(s)=\sum_{i=0}^{n^2}c_iy_i(s)$$

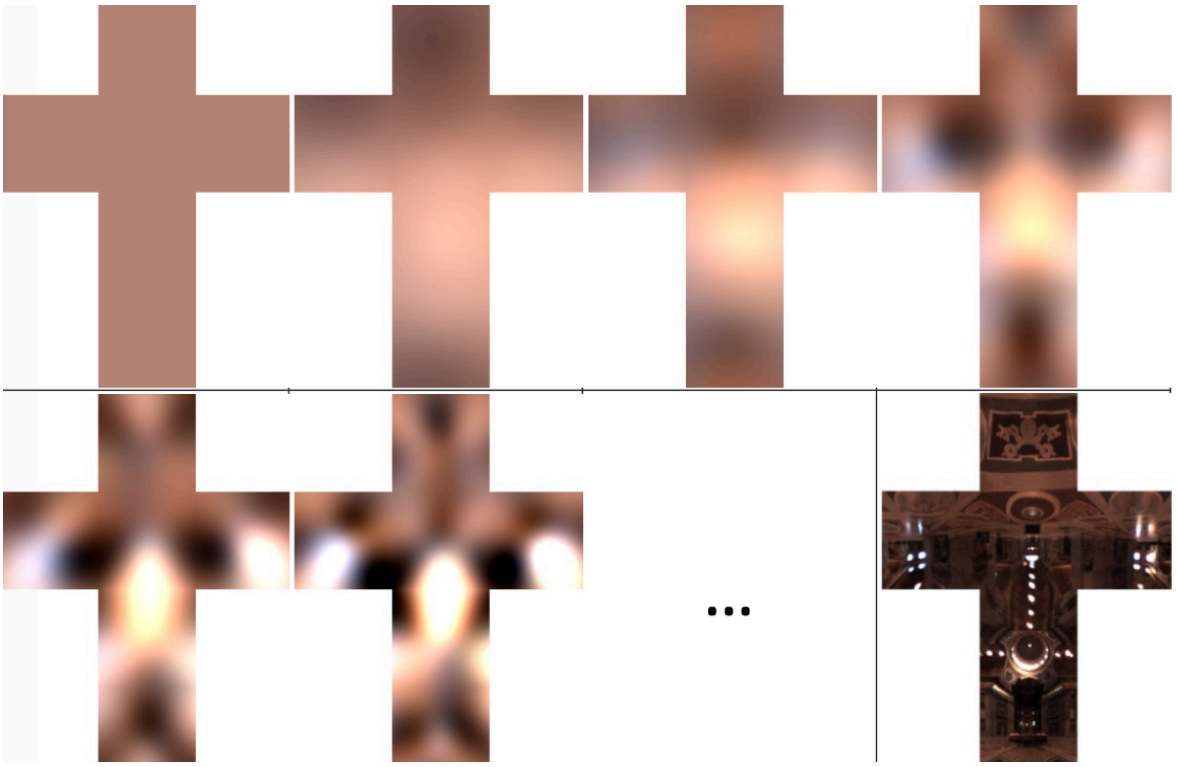
其中阶数越高，重构的结果越趋近于原函数。

我们求解代码如下：

```
//计算系数
void Harmonics::Evaluate(const std::vector<Vertex>& vertices)
{
    int n = (degree_ + 1)*(degree_ + 1);
    coefs = vector<glm::vec3>(n,glm::vec3(0.0));

    //对球面上的所有采样点进行积分
    for (const Vertex& v : vertices)
    {
        vector<float> Y = Basis(v.theta,v.phi);
        for (int i = 0; i < n; i++)
        {
            //v.color是我们从原函数f(s)中采样到的颜色
            coefs[i] = coefs[i] + Y[i] * v.color;
        }
```

```
    }
    for (glm::vec3& coef : coefs)
    {
        coef = 4.0f*PI*coef / (float)vertices.size();
    }
}


//用系数重构f(x)
glm::vec3 Harmonics::Render(const double theta,const double phi)
{
    int n = (degree_ + 1)*(degree_ + 1);

    vector<float> Y = Basis(theta,phi);
    glm::vec3 color=glm::vec3(0.0f);
    for (int i = 0; i < n; i++)
    {
        color = color + Y[i] * coefs[i];
    }
    return color;
}

vector<float> Harmonics::Basis(const double theta,const double phi)
{
    int n = (degree_ + 1)*(degree_ + 1);
    vector<float> Y(n);

    for(int l=0 ; l<=degree_; l++){
        for(int m = -1 * l; m <= l; m++){
            //利用索引:n=l(l+1)+m
            Y[l*(l+1)+m] = y(l, m, theta, phi);
        }
    }
    return Y;
}
```

随着阶数提高，效果如下:



好处

将球面上的函数表示为 **正交基的线性组合**，尤其是 **球谐函数（Spherical Harmonics）** 作为正交基，带来了很多好处。具体来说，这种表示方式有助于处理球面上的复杂函数，尤其在计算机图形学、物理模拟、信号处理等领域中，具有非常广泛的应用。下面详细介绍其几个主要好处：

1. 简洁的表示和分解

- **线性组合**：任何球面上的函数都可以通过一组球谐函数的线性组合进行表示。这意味着复杂的球面函数可以被分解为多个不同频率的分量，每个分量由不同阶次的球谐函数表示。
- **频域分析**：球谐函数在球面上类似于傅里叶级数在一维中的作用。通过使用球谐函数，可以将复杂的球面函数分解成低频分量（平滑的部分）和高频分量（细节部分）。这种分解方式使得我们可以分别处理函数的不同部分，更好地理解和操作球面上的数据。

2. 正交性和简化计算

- **正交性**：球谐函数是正交的，这意味着它们彼此独立，能够在空间中良好地分离不同的方向分量。这使得在投影或重建函数时，系数的计算非常简单，可以通过内积操作直接得到系数。
- **简化的投影与重建**：正交基的一个好处是，投影一个函数到这些基上非常简单，通过计算函数和各个球谐基的内积即可得到对应的系数。重建时，只需要对系数进行加权求和。这种操作可以有效减少计算复杂度。

3. 有效处理方向依赖的函数

- **球面上的方向性问题**：许多物理现象或图形学中的光照模型都依赖于方向。球谐函数作为定义在球面上的函数，特别适合处理这些依赖方向的现象。
- **应用于光照建模**：在图形学中，球谐函数常被用于表示全局光照中的环境光照。环境光通常是方向相关的，通过使用低阶的球谐函数，能够非常高效地表示环境光照，而不需要对每个方向进行采样。

4. 平滑逼近

- **低频信息与平滑效果**：低阶的球谐函数可以很好地表示平滑、低频的变化，比如物体的漫反射光照或是平滑的颜色变化。通过使用较低阶次的球谐函数，可以获得平滑的近似，而高阶次的分量则捕捉细节和快速变化的特征。
- **逐级逼近**：可以通过增加球谐函数的阶次逐步逼近复杂的球面函数。最开始使用低阶函数可以很好地表示平滑区域，后续再加入高阶分量捕捉细节信息。这种逐级逼近的特性非常适合多层次优化或压缩算法。

5. 降维与压缩

- **数据压缩**：由于球谐函数的阶次可以控制，使用低阶球谐函数时可以有效压缩数据量。例如，只保留前几阶的球谐系数就可以近似表示原始的复杂球面函数。这种方法可以大幅度降低数据存储和传输的开销，尤其在需要处理大量方向相关的数据时（如光照、反射等），球谐函数常用于数据压缩。
- **对抗噪声的能力**：低阶球谐函数的平滑特性可以帮助消除噪声。如果一个球面函数含有大量噪声，那么高频分量往往对应这些噪声，通过仅保留低频分量，可以去除高频噪声，得到平滑的结果。

6. 在图形学中的应用

- **光照模型**：球谐函数被广泛用于 **全局光照（Global Illumination）** 模型中，用于逼近和表示复杂的光照分布。通过球谐函数，可以简化计算光照时的方向相关性，减少采样的复杂性。
- **动态光照和漫反射**：球谐函数常用于模拟低频的光照现象，尤其是漫反射光照。由于漫反射光是方向相关的，球谐函数的低阶分量可以很好地捕捉这些平滑的光照变化。使用球谐函数表示光照时，可以实现快速的环境光计算，同时获得平滑的光照效果。
- **环境贴图**：球谐函数常被用于压缩和表示环境贴图中的光照数据，通过球谐系数，可以以较少的数据逼近复杂的环境光照效果。

总结

使用 **正交基的线性组合**，特别是球谐函数来表示球面上的函数，带来了以下几个好处：

- 简洁地表示复杂的方向相关函数，并可以分解成不同频率的分量。
- 利用正交性简化了计算，方便进行投影和重建。
- 适合表示和处理方向依赖的光照模型，尤其是在图形学中。
- 提供了逐步逼近的机制，可以通过增加阶次逐步提高逼近精度。
- 带来压缩数据和降噪的能力，在存储和处理方向相关数据时具有显著优势。
- 利用球面原始函数，可以用低阶去模拟环境光的效果，高阶则可以用来模拟镜面反射等高精度效果。