# Surface Normal Vector in OpenCascade

eryar@163.com

摘要 Abstract：表面上某一点的法向量（Normal Vector）指的是在该点处与表面垂直的方向。对于平面，其上各点的法向是一样的，统一为这个平面的法向。对于曲面，各点具有不同的法向量。几何对象的法向量定义了它在空间中的方向，法向量是在进行光照处理时的重要参数。所以在显示造型算法离散曲面后的网格时，设置正确的法向量对场景的光照、光线追踪效果有直接影响。本文结合 OpenCascade 中代码，对其法向量的计算方法进行分析，稍加修改即可用到实际的程序中。

关键字 Key Words：OpenCascade, Normal Vector, Mesh Normal, OpenSceneGraph,

## 一、引言 Introduction

表面上某一点的法向量（Normal Vector）指的是在该点处与表面垂直的方向。对于平面，其上各点的法向是一样的，统一为这个平面的法向。对于曲面，因为它在计算机图形中是由许多片小平面的多边形逼近来表示的，所以每个顶点的法向量都不一样。因此，曲面上每个点的法向量计算就可以根据不同的应用有不同的算法，则最后的显示效果也是不同的。几何对象的法向量定义了它在空间中的方向，法向量是在进行光照处理时的重要参数。因为法向量决定了该如何计算光照，决定了该点能够吸收多少光照。

OpenGL 有很大的灵活性，它只提供赋予当前顶点法向量的函数，并不在内部具体计算其法向量，这个值由编程者自己根据需要设置。尽管法向量并不需要指定为单位向量，但是如果所有表面法向量都使用单位法向量可减少计算量。使用下列命令可自动将所有非单位法向量单位化：glEnable(GL_NORMALIZE)，该命令也会对那些经过缩放或错切等几何变换的表面向量进行规范化。另一可用选项是指定一个法向量列表，与顶点数组混合使用。

在很多应用程序中网格上的各顶点都需要一个表面法向量，它的用途很广泛：

- 计算光照；
- 背面剔除；
- 模拟粒子系统在表面的“弹跳”效果；
- 对只需要正面而加速碰撞检测；

通常我们在绘制几何体时都会指定法向量。当得到一个模型本身没有法向量时，则有必要通过现有的数据生成。通常表面法向量可能保存于三角形级或顶点级，其中的一个技巧就是平均相邻三角形的表面法向量，并将结果规范化。一般可以这样假设三角形的顶点按逆时针排列，通过叉乘就可以得到外表面的法向量了。当然有些有情况下，顶点的顺序是未知且比较混乱的，这样就比较麻烦了。这个笔者也没有仔细深入研究，推荐读者看一下《计算非固定结构序列的多边形的顶点法线》这篇论文。通过平均三角形法向量求得顶点法向量是一种经验性的方法，不具有通用性，虽然很多情况下可以正确地工作，但有些情况下还是无法正常使用的。（以上内容摘自《OpenSceneGraph 三维渲染引擎编程指南》）

二、计算法向量 Finding Surface Normal Vectors

OpenGL 并不能自动计算几何对象的法向量，而只能由用户显式指定。法向量的计算是一个纯粹的几何和数学问题，这里只简略地区分了几种情况。

2.1 计算平面的法向量

首先，讲述平面法向量的计算方法。在平面内，有两条相交的线段，假设其中一条为矢量 W，另一条为矢量 V，且平面法向量为 N。如图 2.1 所示，则平面法向量就等于两个矢量的叉积（遵循右手法则），即 N＝W x V。

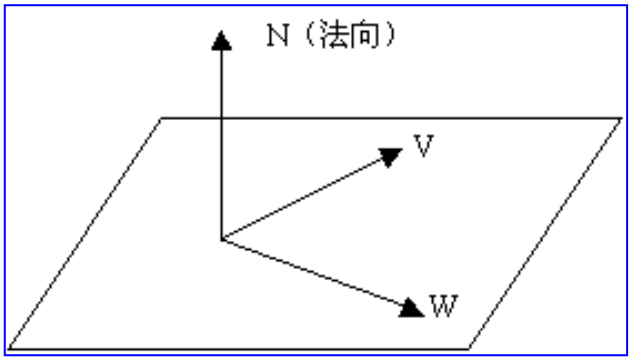


Figure 2.1 Normal Vector of Plane

比如计算一个三角形的法向就可以用它的三个顶点来计算，如图 2.2 所示：

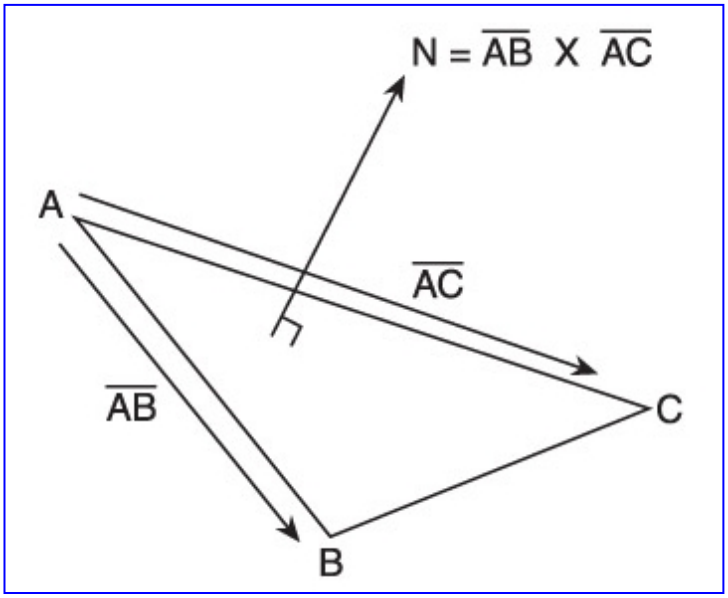


Figure 2.2 Finding the normal vector of a triangle

2.2 计算解析曲面的法向量

解析曲面是由数学方程描述的平滑的、可微曲面。在 OpenCascade 中曲面是由 Geom_Surface 来用参数 u, v 来表示的，相当于曲面的数学方程，是曲面的精确表示。通过计算曲面上一点 u，v 对应的一次微分即可得到曲面在该点处的切线，如下图所示：

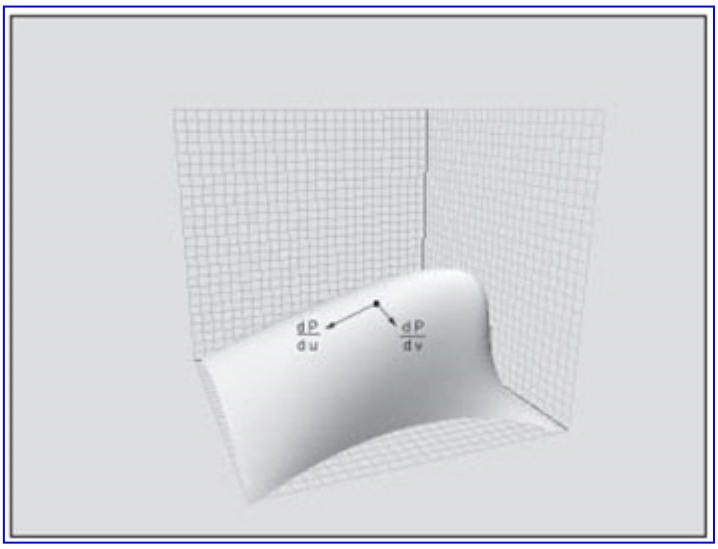


Figure 2.3 Tangents on a surface

关于参数 u，v 表示的 Bezier 曲面的微分计算方法如下所示：

$$\frac{dP}{du} = \sum_{i=0}^{N}\sum_{j=0}^{M} \frac{dB_i(u)}{du} B_j(v) P_{i,j}$$

$$\frac{dP}{dv} = \sum_{i=0}^{N}\sum_{j=0}^{M} B_i(u) \frac{dB_j(v)}{dv} P_{i,j}$$

若需要计算参数对应点处的法向量，还需要对这两个切向量进行叉乘即可，计算方法如下所示：

$$N = \frac{dP}{dv} \times \frac{dP}{du}$$

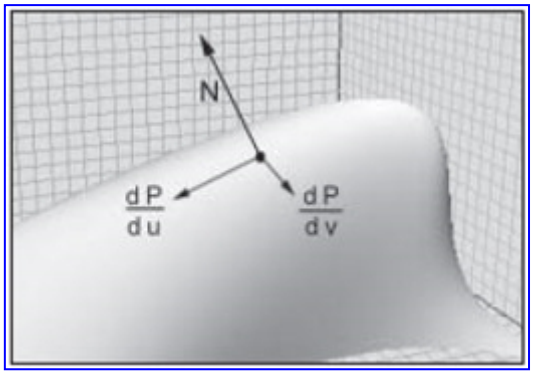


Figure 2.4 Normal on a surface

2.3 计算多边形的法向量

在 OpenGL 中，这种情况占了大多数。求平均多边形的法向量，利用不在同一直线上的多边形三个顶点 v1, v2, v3，则两个矢量的叉积（（v2 - v1）x（v3 - v1））垂直于多边形，即为该多边形的法向量，计算后需要经过规范化处理。

对于求多边形网格上各顶点上的法向量，由于每个顶点同时位于几个不同的多边形边界上，则需要求出周围几个多边形的法向量，然后做加权平均。一般来说，可以使用每个多边形的面积做为加权的权值。

如下图所示，曲面顶点 P 的法向就等于其相邻的四个平面的法向平均值：

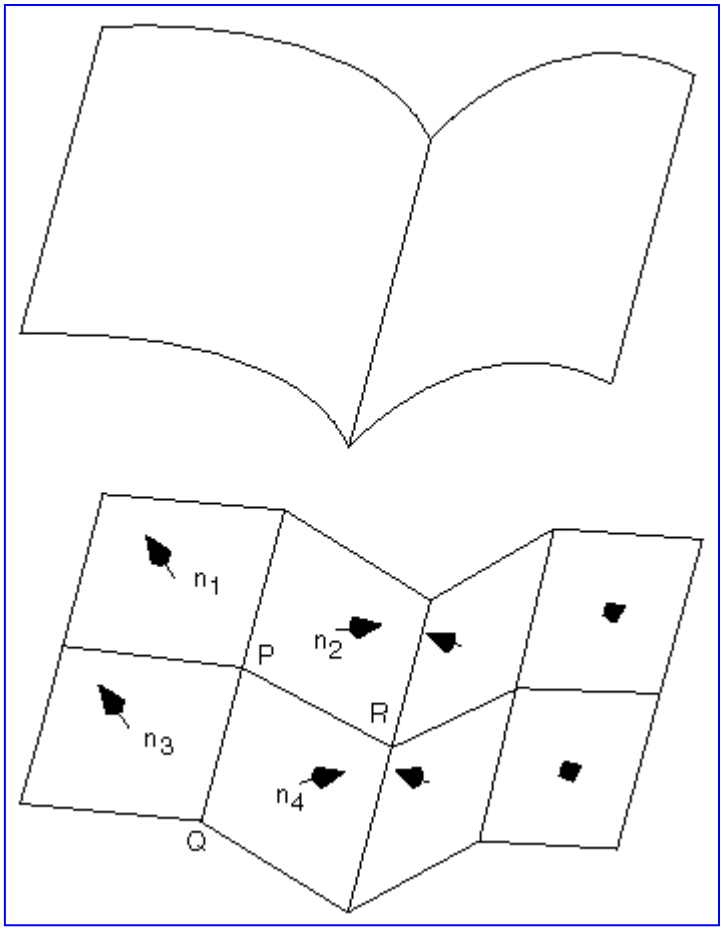


Figure 2. 曲面顶点的平均法向计算

注：当计算多边形网格表示的曲面时，最好是使用平均法向的方法来计算。当曲面是用参数方程来表示时，就可以用求微分和叉乘的方法来直接计算法向量，不再需要使用平均法向了。

三、程序实现 Demo Code

3.1 OpenCascade 中曲面法向量的计算 Compute normal in OpenCascade

在 OpenCascade 中将形状数据保存为 STL 格式时就涉及到了将形状三角剖分及其法向量的计算实现。其计算方法如上所述，也是分成三种方式：

- 参数方程表示的曲面；
- 平面；
- 网格；

将其代码列出如下：

```
//function computes normals for surface
static void Normal(const TopoDS_Face&  aFace,
         Poly_Connect&      pc,
         TColgp_Array1OfDir& Nor)
{
  const Handle(Poly_Triangulation)& T = pc.Triangulation();
  BRepAdaptor_Surface S;
  Standard_Boolean hasUV = T->HasUVNodes();
  Standard_Integer i;
  TopLoc_Location l;
  Handle(Geom_Surface) GS = BRep_Tool::Surface(aFace, l);

  if (hasUV && !GS.IsNull()) {
    Standard_Boolean OK = Standard_True;
    gp_Vec D1U,D1V;
    gp_Vec D2U,D2V,D2UV;
    gp_Pnt P;
    Standard_Real U, V;
    CSLib_DerivativeStatus Status;
    CSLib_NormalStatus NStat;
    S.Initialize(aFace, Standard_False);
    const TColgp_Array1OfPnt2d& UVNodes = T->UVNodes();
    if (!S.GetType() == GeomAbs_Plane) {
      for (i = UVNodes.Lower(); i <= UVNodes.Upper(); i++) {
    U = UVNodes(i).X();
    V = UVNodes(i).Y();
    S.D1(U,V,P,D1U,D1V);
    CSLib::Normal(D1U,D1V,Precision::Angular(),Status,Nor(i));
    if (Status != CSLib_Done) {
      S.D2(U,V,P,D1U,D1V,D2U,D2V,D2UV);

CSLib::Normal(D1U,D1V,D2U,D2V,D2UV,Precision::Angular(),OK,NStat,
Nor(i));
    }
    if (aFace.Orientation() == TopAbs_REVERSED) (Nor(i)).Reverse();
      }
    }
    else {
      gp_Dir NPlane;
      U = UVNodes(UVNodes.Lower()).X();
      V = UVNodes(UVNodes.Lower()).Y();
      S.D1(U,V,P,D1U,D1V);
      CSLib::Normal(D1U,D1V,Precision::Angular(),Status,NPlane);
      if (Status != CSLib_Done) {
    S.D2(U,V,P,D1U,D1V,D2U,D2V,D2UV);
    CSLib::Normal(D1U,D1V,D2U,D2V,D2UV,Precision::Angular(),OK,NSt
at,NPlane);
      }
```

```
      if (aFace.Orientation() == TopAbs_REVERSED) NPlane.Reverse();
      Nor.Init(NPlane);

    }
  }
  else {
    const TColgp_Array1OfPnt& Nodes = T->Nodes();
    Standard_Integer n[3];
    const Poly_Array1OfTriangle& triangles = T->Triangles();

    for (i = Nodes.Lower(); i <= Nodes.Upper(); i++) {
      gp_XYZ eqPlan(0, 0, 0);
      for (pc.Initialize(i);  pc.More(); pc.Next()) {
    triangles(pc.Value()).Get(n[0], n[1], n[2]);
    gp_XYZ v1(Nodes(n[1]).Coord()-Nodes(n[0]).Coord());
    gp_XYZ v2(Nodes(n[2]).Coord()-Nodes(n[1]).Coord());
    eqPlan += (v1^v2).Normalized();
      }
      Nor(i) = gp_Dir(eqPlan);
      if (aFace.Orientation() == TopAbs_REVERSED) (Nor(i)).Reverse();
    }
  }

}
```

如果是参数方程表示的曲面，若不是平面，则根据切线的叉乘来计算各顶点处的法向量；若是平面，则只计算一个顶点处理的法向量，减少计算量。 若是离散后的网格面，则根据三角形的法向量的计算方法来计算每个顶点处的法向量。

3.2 OpenSceneGraph 中网格曲面的法向量计算 Compute normal in OpenSceneGraph

生成顶点法向量（osgUtil::SmoothingVisitor）类继承自 osg::NodeVisitor 类，采用 Visitor 模式，遍历场景中的几何体，生成顶点法向量。osgUtil::SmoothingVisitor 的使用很方便。对算法实现感兴趣的读者可以结合源程序来理解研究。

## 四、结论 Conclusion

OpenCascascade 中有曲面的参数表示，所以对这类曲面可以得用参数方程计算出曲面上的顶点的准确法向量。对于没有参数表示的网格曲面，可以用平均法向量的方法来计算出一个法向量。

OpenSceneGraph 中也有快速计算网格曲面法向量的类 osgUtil::SmoothingVisitor。

## 五、参考资料 References


1. Kelly Dempski, Focus on Curves and Surfaces, Premier Press, 2003
2. 王锐，钱学雷，OpenSceneGraph 三维渲染引擎设计与实践，清华大学出版社
3. 肖鹏，刘更代，徐明亮，OpenSceneGraph 三维渲染引擎编程指南，清华大学出版社