Bump Mapping 原理及在 OpenGL 下的实现*

付 恺 李春霞 杨克俭 李 波

(武汉理工大学 武汉 430063)

摘 要 凹凸映像(bump mapping)作为逐像素光照的一种,能够非常逼真地再现物体表面的细节。文章在对传统光照模型分析的基础上,比较了 Gouraud 光照模型和 Phong 模型。并结合目前主流的 3D 图形硬件,说明了实现 Phong 光照模型的可行性。然后介绍了逐像素光照、法线图、切面空间、Cube-Map 等实现凹凸映像的要素,最后阐述了在 OpenGL 下实现 bump mapping 的步骤。

关键词 纹理映像; 凹凸映像; Cube-Map; 逐像素光照

Abstract: Bump Mapping is one of per-pixel-lighting methods, which can give mush details on the surface of objects. This paper compares Gouraud lighting model with Phong lighting model based on the analysis of traditional lighting models. With the latest graphic hardware, this papar also shows the feasibility of implementing Phong model by hardware. Then it introduces such factors as per-pixel lighting, normal map, tangent space, cube-map and so on. Finally, it illustrates the steps to implement Bump mapping in OpenGL.

Key words: texture-mapping; bump-mapping; cube-map; per-pixel lighting

0 引 言

计算机图形学研究的一个很重要的方向就是最大限度地在显示设备上再现真实的图像,即真实感光照研究,用计算机法生成相片级别的图像。在计算机图形学经过了 20 多年的发展后,相关的技术已经比较成熟,但对于实时图形图像系统来说,由于多数算法相当复杂,特别是对大量浮点数据的处理,因此,很多实时系统不得不采用一些简化的算法和光照模型。由于实时系统使用的是简化模型和算法,是一种对精确模型的近似,所以在某些特殊的情况下,这种近似造成误差会非常大,而 Phong 光照模型有效地避免传统方法的缺陷,本文将介绍相关内容。

1 Bump Mapping 的原理

虽然纹理映像可用于添加精致的表面细节,但它对于模拟粗糙的物体表面,如橘子、草莓和葡萄干等物体则不合适。纹理图案的光照细节通常与场景中的光照方向并不一一对应。生成物体表面凹凸效果的较好方法是在光照模型计算中使用扰动法向量,该技术被称为凹凸映像(bump mapping)。

若 $\boldsymbol{P}(u, v)$ 为一个参数曲面上的点,可以通过计算得到该点处的表面法向量。

$\boldsymbol{N} = \boldsymbol{P}_u \times \boldsymbol{P}_v$ (× 为向量运算中的叉乘)

式中:$\boldsymbol{P}_u$ 和 $\boldsymbol{P}_v$ 是关于参数 u 和 v 的偏导数。为了得到扰动法向量,可以在表面点的向量方向上增加一个小的扰动函数 $b(u,v)$,得到经过扰动后的曲面方程。

$$\boldsymbol{P}(u, v) = \boldsymbol{P}(u, v) + b(u, v)\,\boldsymbol{n}$$

它在表面单位法向量上 $\boldsymbol{n} = \boldsymbol{N}/|\boldsymbol{N}|$ 增加凹凸效果,扰动后的表面法向量为:$\boldsymbol{N} = \boldsymbol{P}_u \times \boldsymbol{P}_v$,然后根据扰动向量的 u 分量来计算偏导数:$\boldsymbol{P}_u = \boldsymbol{P}_u + b_u\boldsymbol{n} + b\boldsymbol{n}_u$

假定凹凸函数 b 很小,则可以忽略最后一项得到 $\boldsymbol{P}_u \approx \boldsymbol{P}_u + b_u\boldsymbol{n}$;同样:$\boldsymbol{P}_v \approx \boldsymbol{P}_v + b_v\boldsymbol{n}$。

则被扰动点处的物体表面法向量为

$$\boldsymbol{N} = \boldsymbol{P}_u \times \boldsymbol{P}_v + b_v(\boldsymbol{P}_u \times \boldsymbol{n}) + b_u(\boldsymbol{n} \times \boldsymbol{P}_v) + b_u b_v(\boldsymbol{n} \times \boldsymbol{n}) \quad (1)$$

因为 $\boldsymbol{n} \times \boldsymbol{n} = 0$,故

$$\boldsymbol{N} = \boldsymbol{N} + b_v(\boldsymbol{P}_u \times \boldsymbol{n}) + b_u(\boldsymbol{n} \times \boldsymbol{P}_v) \quad (2)$$

3 光照模型

在 OpenGL 中,实体渲染方式有三角形(trian-

收稿日期:2003-10-03 修改稿收稿日期:2004-02-12
* 国家自然科学基金项目资助(60073057)

gles)，三角带(triangle stripes)和三角形扇面(triangle fans)，其它非三角形实体也都是转换为三角形后再渲染的。当采用三角形方式时，OpenGL 会认为客户端每调用 3 次 glVertex 就构成一个三角形；当采用三角带方式时，OpenGL 会认为客户端每调用一次 glVertex 就会和之前调用的两次 glVertex所给出的顶点构成一个三角形。三角形组合完成后，应进行像素处理，Gouraud 模型的像素处理是在一个三角形内部进行颜色插值，最后渲染出像素的颜色就是这里插值产生的颜色。

Gouraud 光照模型是简单的局部光照模型，使用硬件非常容易实现，所以 Gouraud 模型得到绝大多数硬件的支持。这种模型的缺点是会产生很明显的"马赫带"效果，这种现象在三角形的密度比较稀疏时表现得尤为明显。Blinn 提出的一种近似的 Phong 的简化模型，改变了 Gouraud 模型的缺陷，视觉效果要比 Gouraud 真实得多。

计算每个顶点，采用的是 Blinn 提出的 Phong 模型简化计算方法，即光源相对顶点的位置(L)，光源离顶点的距离(d)，光源的环境光(*ambient*)，漫射光(*diffuse*)，反射光(*specular*)和自发光(*emission*)的属性(强度，颜色)都将在计算中考虑。但实际进行逐像素光照计算时，由于受目前硬件发展水平的限制，要实现实时渲染，不能考虑所有影响因素，需进行简化。在忽略模型的自发光、环境光、聚光和光的衰减影响情况下，其光照计算公式如下：

$$out_{col} = diffuse_{col} \cdot \max\{\boldsymbol{N} \cdot \boldsymbol{L}, 0\} + spec_{col} \times (\max\{\boldsymbol{H} \times \boldsymbol{N}, 0\})^{m} \quad (3)$$

式中：out_{col} 为输出颜色；$diffuse_{col}$ 为材质和光源的漫反射颜色之积；$spec_{col}$ 为材质和光源的镜面光颜色之积；$\boldsymbol{N}$ 为顶点法线向量；$\boldsymbol{L}$ 为入射光向量；$\boldsymbol{V}$ 为顶点到视点的向量；$\boldsymbol{H}$ 为 $\boldsymbol{L}$ 和 $\boldsymbol{V}$ 间的半角向量；m 为镜面反射指数，如图 1 所示。

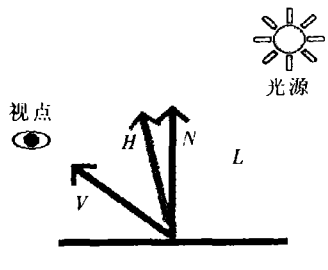

图 1 光照

虽然每个顶点光照使用的是简化 Phong 模型，但最后显示的图像是基于像素的，每个三角形内部像素的颜色是通过其顶点颜色插值(interplantation)得到的。考虑下述情况：当某点光源紧靠近比较大的三角形的重心时，由于顶点离点光源比其离三角形的重心远，所以其顶点处的光强应小于其中心处的光强，但使用插值方式来计算像素的颜色时，由于只针对顶点使用的 phong 模型，所以会得到如图 2 所示的失真的结果(线框标识为点光源的位置)，而正确的光照结果应如图 3 所示(逐像素计算得到)。

图 2 失真的光照

图 3 逐像素光照

2 逐像素光照以及 Bump Mapping

要得到逐像素光照，需对每个像素使用 Phong 模型来计算，即需得到每个像素光线的入射方向(L)，像素的法线方向(N)和光源的光强以及光源离像素的位置等信息，计算中需对向量进行点乘运算，所以其计算的代价也是非常大的。由于目前主流硬件在功能上的限制，这里仅考虑 *Blinn* 模型，即对于每个像素的计算，考虑使用公式(3)。

以上所用的这些向量都是在切面空间中考虑的，切面空间也叫做表面局部空间或者纹理空间[1]。如图 4 所示，切面空间用坐标轴 $\boldsymbol{T}$，$\boldsymbol{B}$，$\boldsymbol{N}$ 表示。以点 O 为例，$\boldsymbol{N}$ 代表点 O 上的法线方向，$\boldsymbol{T}$ 代表点 O 上的切线方向，$\boldsymbol{B}$ 则通过 $\boldsymbol{N}$，$\boldsymbol{T}$ 叉乘得到，即 $\boldsymbol{B} = \boldsymbol{N} \times \boldsymbol{T}$，如图 4 所示(图片来自文献[2])。

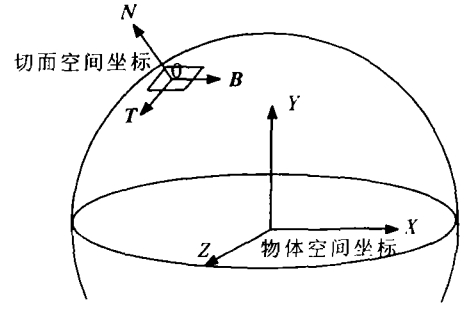

图 4 切面空间坐标

有了向量 $\boldsymbol{T}$，$\boldsymbol{B}$，$\boldsymbol{N}$，就可以构造 $\boldsymbol{TBN}$ 矩阵，$\boldsymbol{TBN}$ 矩阵用来把顶点在对象空间的坐标表示转换为在切面空间的坐标表示

$$\boldsymbol{V}_{\text{tangent space}} = \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{bmatrix} \boldsymbol{V}_{\text{object space}} \quad (4)$$

这个矩阵就是模型坐标到模型顶点的切面空间的坐标。要在 Blinn 模型的基础上实现逐像素的光照,就要求每个像素处的光照方向(L)、每个像素处的法线(N),照相机的观察方向(V),且 L,N 和 V 必须是在同一坐标空间中定义的。这里的实现方法是使用法线图(Normal Map)来记录像素的法线方向,使用 Cube Map 来记录光源的入射方向,在计算镜面光时需要考虑照相机的观察方向(V),每个顶点的 $H = 1/2 \cdot (V + L)$。三角形面片内的像素的 L 和 H 都是通过 Cube Map 插值得到的。

所谓 Cube-Map 是这样一种特殊的纹理对象:通常的纹理映像是针对二维图像的,纹理坐标也是二维的。Cube-Map 是对二维纹理的一种扩展,它使用前、后、左、右、上、下 6 个二维的图像来表现与之对应方向的景物,并且使用含有 S、T、R 3 个分量的纹理坐标来进行纹理取样。在实现 Bump-Maping 时,Cube-Map 的一个重要用途就是储存光源的方向信息。对 CubeMap 进行取样需要使用 3 个分量的向量,在 OpenGL 中就意味着需要使用 glTexCoord3f 而不是 glTexCoord2f。CubeMap 可以这样理解:假设 CubeMap 四面体的中心在世界坐标的原点,给定一个纹理坐标(S、T、R),就可以根据此向量从原点引出一条射线,射线与 CubeMap 四面体的交点就是纹理坐标(S, T, R) 的取样点了,这也就是说(1, 2, 1)和(2, 4, 2)取出的像素的颜色相同,依此类推,(S, T, R) 和(kS, kT, kR) 取出的像素相同 。

逐像素光照的意义在于:对每个像素进行点乘运算。向量的点乘运算是计算光照强度所必需的计算,比如:计算 Diffuse 光时,就需要把光源的方向 L 点乘顶点(或像素)法线的方向 N,得到的标量就是此顶点(或像素)的 Diffuse 光强度。在早期的硬件上,由于功能的限制,无法实现针对逐像素的点乘计算,所以无法真正实现逐像素的光照。目前主流的显示芯片,只要具有支持 DOT_PRODUCT3 的功能,还是可以在某种程度上进行逐像素光照处理的。

4 BumpMapping 在 OpenGL 中的实现

Bump Maping 是逐像素光照的一种,其关键在于点乘的逐像素计算。在基于 OpenGL1.1 规范下,要实现逐像素的点乘计算,一般需要使用以下扩展:①ARB_texture_cube_map,提供对 Cube Map 的支持;②GL_ARB_multitexture。提供多纹理的支持,后面将将会说明这个扩展的必要性;③GL_ARB_texture_env_combine 和 GL_ARB_texture_env_dot3。这两种扩展在 GL_ARB_multitexture 的基础上提供了对不同纹理的片元进行运算的功能,GL_ARB_texture_env_dot3 提供了对两个不同纹理层级(texture stage)进行点乘的功能。

模型的纹理贴图需要有:①Diffuse 纹理贴图;②模型面片的法线图(normal map)。如果要实现更复杂一些的效果,如各向异性光照(anisotropic lighting),材质在一个三角面中的变化等,就需要有更多的纹理贴图,在这里只实现最基本的逐像素光照。对于模型的每一个顶点需要以下信息:① 顶点的位置(P);② 顶点的法线(N),③顶点的 Diffuse 纹理贴图的纹理坐标;④光源的入射方向(L)或顶点的 H;⑤顶点 Normal Map 中的纹理坐标;⑥顶点切平面空间的 S 和 T 轴在模型坐标中的向量,在这些顶点信息中,⑤和⑥是在传统的光照模型(gourand)计算中所不需要的,但对于 S 和 T 理论上只需要给出一个就可以了,因为另外一个可以通过给出的 S 或 T 与顶点法线叉乘得到,在这里,同时给出 S 和 T 以提高实时的绘制速度,即帧数,其实,对于任何静态的模型都可以通过此方法提高绘制的效率。顶点 Diffuse 贴图的纹理坐标和 Normal Map 的纹理坐标可以是一样的,这取决于模型的构造方式。给出 L 的目的是:使用 L 作为 Cube Map 的纹理坐标从中取样,在光栅化时,每个像素从此 Cube Map 中取出纹理坐标就是此像素处的光照方向(L),在计算镜面光时 H 的计算也类似,不同的是所给出的顶点信息不是 L 而是 H,有了像素处的 L 或 H,需要的就是像素处的 N,N 的计算方式就是从 Normal Map 中取样,因为 Normal Map 的纹理坐标已经在顶点信息中给出,所以其取样方式和一般的二维纹理取样无任何区别,有了像素处的 $L(H)$ 和 N,就可以通过点乘运算得到像素处的光强,接下来要做的就和传统的方法没什么区别,主要工作是像素处的光强乘以像素处的 Diffuse 颜色,Diffuse 颜色可以通过顶点颜色插值,也可以通过纹理贴图得到,最后考虑的就是模型的材质等信息,把像素的颜色写到帧缓冲区。

以上是在 OpenGL 实现 Bump Mapping 的具体步骤,可以看出,至少需要两个同时纹理(simultaneoustexture),即 Cube Map 和 Normal Map,所

基于 Hough 变换的车牌倾斜检测算法

包　明　路小波

（东南大学　南京 210096）

摘　要　典型的角度检测算法是进行 Hough 变换后寻找最长直线的倾斜角度，一般情况下，待处理图象中未必有明显的较长直线，甚至存在一些对图像进行正确检测的长直线干扰。文章介绍了一种基于统计考虑的数据分析思路，以期在基于 Hough 变换的角度检测算法上取得更高的准确性。

关键词　车牌识别；角度矫正；Hough 变换

Abstract: During vehicle license plates recognition and some other image processing programs, it is important to rotate the angle of the image. The traditional methods to search the lean of a picture is based on HOUGH-transform(HT). Here we will specify a method based on HT. The key is how to analyze the data getting form HT. It is not very efficient if we just follow the method in the books, since there are not such a long line in the image to show the angle of the image and even the angle of the longest line may be a counteraction to find the real lean. Here, we will consider these problems using statistics in order to raise the quantity of angle searching.

Key words: plates recognition; angle rotate; hough-transform

0　引　言

目前的车牌识别算法，一般分两步进行：首先对车牌进行定位和分割；然后对分割后的车牌进行字符识别。在第一步中，待处理图像的倾斜，往往成为精确定位和分割的一大障碍。传统的通过用 Hough 变换寻找图像最长直线的角度以确定图像倾斜度的方法[1,2]，在实际运用中并不准确有效：事实上，笔者并未根据图像的实际情况充分利用Hough变换域中的数据，而是经过分析和改进后，提出一种基于 Hough 变换的更具实际使用意义的图像倾斜检测算法。

1　基于 Hough 变换的算法

对于二维数字图像 F，我们一般用笛卡儿直角坐标系按每个离散像素的位置(x,y) 表示其灰度 G(颜色)[2,3]；对于图象上每个点，有

$$G = F(x,y)$$

对于图像上一条直线，可以斜(斜率)截(截距)式表示为

以多纹理的支持是必须的。考虑到通用性，这里的实现只使用了 ARB 的扩展，但是因为具体硬件厂商的扩展能提供更多的功能，所以目前使用具体硬件厂商提供的扩展能实现更好的效果，比如可以用 nVidia 公司的 NV_register_combine 等。

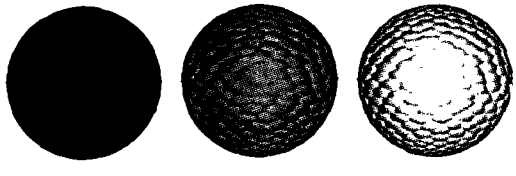

图 5　按照 Blinn 光照模型实现 Bump Mapping 的 Demo

按照 Blinn 光照模型实现 Bump Mapping 的 Demo 如图 5 所示(左边的球是一般纹理贴图，中间的球加了凹凸映像，右边的球再加入了镜面光)，在 CPU 为 AMD althon 2000 + XP，显卡为 Nvidia GeForce Ti500 的 PC 机上，每秒的帧数为 315，完全达到了实时的速度。

收稿日期：2003-12-15

参考文献

1　Donald Hearn M, Pauline Baker. 北京：电子工业出版社，1998. 417 ~ 421

2　朱腾辉，刘学慧，吴恩华. 基于像素的光照计算技术. 计算机辅助设计与图形学学报，2002(9)：861 ~ 865

3　王玉华，杨克俭，曾梅兰. 面向对象技术在三维虚拟场景建模中的应用研究. 武汉理工大学学报(交通科学与工程版)，2003(1)：37 ~ 39